Report Number: WUCS-93-19

1993-01-01

Approximation Algorithms for Configuring Hierarchical Nonblocking Communication Networks

J. Andrew Fingerhut

A framework is given for specifying nonblocking traffic limits in a connection-oriented communications network. In this framework, connections may be point-to-point or mutlipoint, and the data rates may vary from one connection to another. The traffic limits may be "flat", or they may also be hierarchical, representing communities of interest within the network that have higher traffic among themselves than with the rest of the network. The communication networks are constructed from switches (or nodes) and trunks, which connect pairs of switches. This framework is intended to model Asynchronous Transfer Mode (ATM) networks and traffic. We present a way... **Read complete abstract on page 2.**

Approximation Algorithms for Configuring Hierarchical Nonblocking Communication Networks

J. Andrew Fingerhut

Complete Abstract:

A framework is given for specifying nonblocking traffic limits in a connection-oriented communications network. In this framework, connections may be point-to-point or mutlipoint, and the data rates may vary from one connection to another. The traffic limits may be "flat", or they may also be hierarchical, representing communities of interest within the network that have higher traffic among themselves than with the rest of the network. The communication networks are constructed from switches (or nodes) and trunks, which connect pairs of switches. This framework is intended to model Asynchronous Transfer Mode (ATM) networks and traffic. We present a way of computing a lower bound on the cost of any nonblocking network that can satisfy the limits, and show that this lower bound is good analytically and through experiments on randomly generated instances.

Approximation Algorithms for Configuring Hierarchical Nonblocking Communication Networks

J. Andrew Fingerhut

WUCS-93-19

August 1993

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130-4899

This work is supported by the National Science Foundation, Ascom Timeplex, Bell Communications Research, Bell Northern Research, Gold Star Communications, Italtel SIT, NEC America, NTT and SynOptics.

August 26, 1993

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

A framework is given for specifying nonblocking traffic limits in a connection-oriented communications network. In this framework, connections may be point-to-point or multipoint, and the data rates may vary from one connection to another. The traffic limits may be "flat", or they may also be hierarchical, representing communities of interest within the network that have higher traffic among themselves than with the rest of the network. The communication networks are constructed from switches (or nodes) and trunks, which connect pairs of switches. This framework is intended to model Aynchronous Transfer Mode (ATM) networks and traffic. We present a way of computing a lower bound on the cost of any nonblocking network that can satisfy the limits, and show that this lower bound is good analytically and through experiments on randomly generated instances.

This work is supported by the National Science Foundation, Ascom Timeplex, Bell Communications Research, Bell Northern Research, Goldstar, Italtel SIT, NEC, NTT, and SynOptics.

1. Introduction

There has been much work done on algorithms for configuring communication networks as cheaply as possible, given that they must satisfy certain offered traffic. For example, see the surveys by Minoux [Min89] and Magnanti and Wong [MW84]. However, most of the work in this area assumes that the network designer knows, at least approximately, how much traffic will appear between every pair of nodes in the network. This is reasonable for very large networks, such as the United States telephone network, for which the aggregate traffic is so large that those quantities are stable, and data is available to make such estimates.

For local area networks up to campus-sized and possibly metropolitan area networks, the traffic quantity is smaller, and the magnitude of traffic between node pairs is often much less stable over short time periods, and harder to estimate or predict. In this paper we present a way of specifying traffic which would be easier for a network designer to estimate, based on the total traffic that a node may initiate or receive at one time, called the source termination capacity and destination termination capacity. No traffic matrix is necessary, as the traffic may go between any pairs of nodes so long as no node violates their termination capacity restrictions. A traffic matrix may be specified to restrict the traffic further (called point-to-point restrictions here).

Given this framework, we analyze the performance of a simple network configuration algorithm, both experimentally and analytically. This algorithm simply finds the cheapest star network among the n such possible networks, where n is the number of nodes. An efficient algorithm for computing a lower bound on the cost of any feasible network is developed as well. The experiments and analysis show that this simple algorithm very often produces networks which are within a few percent of optimal.

The traffic limits are generalized so that a network designer may model "clusters" of nodes which have high traffic among themselves, but smaller traffic to the rest of the network. Such traffic patterns are common in most communication networks which are larger than a small local area network. The simple configuration algorithm and lower bound computation algorithm are generalized for these traffic limits as well, and experimental results show that solutions for randomly generated instances are usually within 10% of optimum.

This work is intended for configuring ATM networks. These are connection-oriented, where different connections may have different rates, and connections may be multicast, involving more than two nodes. We explicitly account for these properties in our work, and the networks produced will never block a request for a new connection, so long as the request does not violate the specified traffic limits.

In section 2 we define networks and connections, nonblocking networks, and the nonblocking network configuration problem. In section 3 we define flat (as opposed to hierarchical) traffic limits, give an efficient algorithm for computing a lower bound on the cost of any nonblocking network, develop an algorithm for link dimensioning in a tree network, present experimental results of randomly generated instances with flat traffic limits, and prove two theorems which were motivated by the experiments. In section 4 we generalize the flat traffic limits to hierarchical, modify the algorithms for computing the lower bound and link dimensions, and present further experimental results. In section 5 we generalize

the link dimensioning algorithm for networks with arbitrary topologies, and verify that all of the previous link dimensioning algorithms are correct when multipoint traffic is also allowed.

2. Definitions

2.1. Networks, connection requests, and connections

A *network* $\mathcal{N}$ is a directed graph with node set N, link set L, and a function $cap : L \rightarrow \mathbb{N}$, where $cap(l)$ is the bandwidth, or capacity, of link l. $\mathbb{N}$ denotes the set of natural numbers.

A *connection request* (or *call*) is a triple $q = (src, dest, rate)$, where the nodes $src(q) \subseteq N$ are the sources of information flow, nodes $dest(q) \subseteq N$ are the destinations of flow, and $rate(q) \in \mathbb{N}$ is the desired rate of the connection. Either or both of the sets $src(q), dest(q)$ must contain exactly one node. If both do, then the request is called *point-to-point.* If $|dest(q)| > 1$, then the request is called a *one to many multipoint request.* If $|src(q)| > 1$, then the request is called a *many to one multipoint request.*

A *connection* (or *route*) is a pair $r = (links, rate)$. $Links(r)$ is a set of links in L which forms a tree. That is, if all of the edges $links(r)$ are considered as undirected edges, then the edges in $links(r)$ with their incident nodes form a connected, acyclic graph. A *point-to-point* connection is a directed path from a source node to a destination node. A *one to many multipoint* connection is a tree with the source node as a root, destination nodes as either leaves or internal nodes of the tree, and all edges directed away from the source. A *many to one multipoint* connection is a tree with the destination node as a root, source nodes as either leaves or internal nodes of the tree, and all edges directed towards the destination. $rate(r)$ is the bandwidth which is used on each link of the connection.

A connection r *realizes a request* q under one of several conditions. If q is point-to-point, then r realizes q if $links(r)$ is a directed path from $src(q)$ to $dest(q)$. If q is a one to many multipoint request, then r realizes q if $links(r)$ is a directed tree with root $src(q)$, all links are directed away from the root, all $dest(q)$ are in the tree, and all leaves are contained in $dest(q)$. If q is a many to one multipoint request, then r realizes q if $links(r)$ is a directed tree with root $dest(q)$, all links are directed toward the root, all $src(q)$ are in the tree, and all leaves are contained in $src(q)$. In all cases, $rate(r) = rate(q)$ should also hold.

A *state* $\mathcal{S}$ of the network is a (multi)set of connections. Given a network $\mathcal{N} = (N, L, cap)$ and a state $\mathcal{S}$, we define the *usage* and *available capacity* of a link l in state $\mathcal{S}$ to be

$$\begin{aligned} usage(l, \mathcal{S}) &= \sum_{r \in \mathcal{S}, r \text{ uses link } l} rate(r) \\ avail(l, \mathcal{S}) &= cap(l) - usage(l, \mathcal{S}) \end{aligned}$$

A state $\mathcal{S}$ is *legal for network* $\mathcal{N}$ if

$$(\forall l \in L)\ (usage(l, \mathcal{S}) \leq cap(l))$$

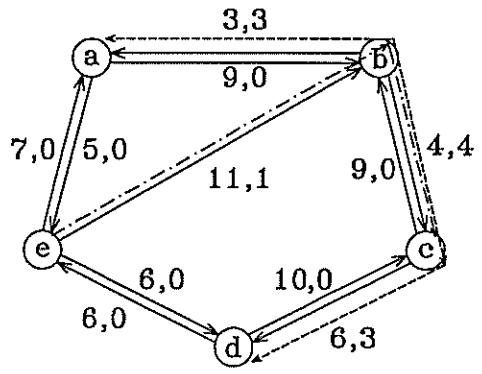

Figure 1: An example state

In other words, every link is used in connections that have a total rate which is at most the link's capacity.

For example, the point-to-point request $(e, c, 1)$ is realized by the connection $(\{(e, b), (b, c)\}, 1)$, which is the dash-dotted path superimposed on the network in Figure 1. Similarly, the one to many multipoint request $(b, \{a, d\}, 3)$ is realized by the connection $(\{(b, a), (b, c), (c, d)\}, 3)$, which is the dashed tree in Figure 1. The links are labeled with two numbers, first capacity and then usage.

2.2. Fixed path routing and nonblocking networks

A routing algorithm is a method for taking a connection request and the current state of the network and determining a connection which satisfies the request. The new connection should not exceed any of the link capacities when it is added to the current state.

In this paper, we consider one simple routing algorithm, called *fixed path* routing. All of the definitions in this section have been simplified with this routing algorithm in mind. See Fingerhut [Fin91, Fin92] for definitions of other routing algorithms, and a more general definition of a nonblocking network.

When fixed path (FP) routing is used, there is a table *path*. For each ordered pair of endpoints u, v, $path(u, v)$ is a directed path from u to v. Whenever a point-to-point connection request $q = (u, v, rate)$ is given, routing algorithm FP returns the path $path(u, v)$. Note that this path may not have enough bandwidth available to handle the new connection, and this will cause the request to block, even though other paths from u to v may have enough available capacity. The name of this routing algorithm comes from its behavior: the path to be used from u to v is *fixed* for the lifetime of the network (or at least for a long time). It ignores the current state of the network.

We will define what it means for a set of connection requests $\mathcal{R}$ to be compatible with traffic limits $\mathcal{T}$ in sections 3.1 and 4.1. Intuitively, the traffic limits restrict the number and bandwidth of connection requests which can be in a compatible set $\mathcal{R}$.

Consider a network $\mathcal{N} = (N, L, cap)$, and a sequence of add and drop requests $q_1, \ldots, q_k$. Each q_i is either a request to add a new connection, or to remove a previously added

connection which has not yet been removed. Define $\mathcal{R}_i$ to be the subset of $\{q_1, \ldots, q_i\}$ from which all drop requests and their corresponding add requests have been removed. $\mathcal{R}_i$ is the set of *active* connection requests after request q_i has been completed. Define the sequence $q_1, \ldots, q_k$ to be compatible with traffic limits $\mathcal{T}$ if for all i, $1 \leq i \leq k$, the set $\mathcal{R}_i$ is compatible with $\mathcal{T}$.

The network $\mathcal{N}$ is *blocking* for traffic $\mathcal{T}$ if there exists a sequence of requests $q_1, \ldots, q_k$ compatible with $\mathcal{T}$ such that $q_k = (u, v, r)$ is an add request, and there is less than r units of bandwidth available on the path $path(u, v)$. We say that the request q_k is blocked. $\mathcal{N}$ is *nonblocking* if there is no such request sequence.

Note that this definition of fixed path routing only allows point-to-point requests and connections. In section 5.2 we explain how to extend this routing algorithm to handle multipoint requests as well.

2.3. The Network Configuration Problem

We now describe a computational problem which models the following scenario. A network manager has several switches in given locations, and knows how much it costs to install links of various capacities between these switches. The manager also knows the traffic limits, and wants to know how much bandwidth to install between each pair of switches so that the resulting network is nonblocking with respect to this traffic.

Nonblocking network configuration

> INSTANCE: A set of nodes N. For each node pair u, v, a nondecreasing function $cost_{u,v} : \mathbb{N} \to \mathbb{N}$, where $cost_{u,v}(x)$ is the cost of installing a link of capacity x from u to v. Traffic limits $\mathcal{T}$ are given by (α, ω, μ).
>
> SOLUTION: A capacity $cap(u, v) \in \mathbb{N}$ for each node pair u, v, and some routing algorithm. This assignment of capacity should make the network nonblocking for traffic $\mathcal{T}$ when the routing algorithm is used.
>
> SOLUTION COST: The cost of the network is the sum of the costs of each link:
>
> $$\sum_{u,v \in N} cost_{u,v}(cap(u, v))$$
>
> OBJECT: Find a solution with minimum cost.

3. Flat traffic limits

What are called flat traffic limits in this paper have been examined before by Fingerhut [Fin92]. They are defined again here in this section. Section 4.1 defines a more general type of traffic called hierarchical traffic limits.

3.1. Definition

Intuitively, a nonblocking network is one which, given any legal state which realizes a compatible set of requests, and an additional connection request compatible with the existing ones, we can find a connection which realizes the request such that the resulting state is legal. It should be clear that unless there is some way to restrict the new requests that come in, any network with finite link capacities will eventually be unable to satisfy the new requests. We will restrict new requests by introducing the concept of termination capacity.

Every node v has a *source termination capacity* $\alpha(v) \in \mathbb{N}$ and a *destination termination capacity* $\omega(v) \in \mathbb{N}$. $\alpha(v)$ ($\omega(v)$) is a number which represents the maximum total rate of all connections in which v may be a source (destination). For example, if $\alpha(v) = 5$, then v may simultaneously be a source in connections with rates 1, 2, and 2, but then it could not be a source in any more connections until an existing connection is removed.

The source termination capacity $\alpha(u)$ may be interpreted as the bandwidth of the interface to the switch u from the collection of terminals connected to u (similarly for $\omega(u)$, but in the opposite direction). This interface has a finite bandwidth, and if it is ever all used, then it is impossible for any more connections to be made which have one of those terminals as a source. Note that additional connections may pass *through* the node, but they cannot start there. Also note that when we say a network is nonblocking, we mean that if a new request does not exceed any of the termination capacities, then there will be enough capacity on the links of the network to satisfy the request. From the point of view of a user operating a terminal, a request may block because other users connected to the local node through the same interface may be using all of the termination capacity. A network designer using the algorithms developed here may wish to vary the values of termination capacity to see how it affects the cost of the network. They may also wish to engineer the network to be nonblocking for values of α and ω which are lower than the bandwidths of the interfaces actually present in the network. This could lower the cost of the network, but then the nonblocking guarantee only applies when the network operates within the designed values. Users could try to exceed those termination capacity values, and then blocking may occur. It is possible that some requests like this could succeed, and then other users operating within their termination capacities may block because someone else is "breaking the rules". This idea of a termination capacity was inspired by the "maximum port weight" β used by Melen and Turner [MT89, MT90].

An additional idea which may be useful in lowering the cost of network configurations is *point-to-point restrictions.* They are specified by giving a value $\mu(u, v) \in \mathbb{N}$ for each $u, v \in N$. The value $\mu(u, v)$ is the maximum total rate of all connections that may exist from u to v simultaneously.

An example of flat traffic limits is given in Figure 2.

Given a network $\mathcal{N} = (N, L, cap)$ and a (multi)set of connection requests $\mathcal{R}$, define the *source usage*, *destination usage*, and *point-to-point usage* under requests $\mathcal{R}$ as

$$src\text{-}usage(u, \mathcal{R}) = \sum_{q \in \mathrm{R},\, u \in src(q)} rate(q)$$

u	$\alpha(u)$	$\omega(u)$
a	4	4
b	6	1
c	2	8

		v		
		a	b	c
u	a		1	4
	b	4		6
	c	2	1	

$\mu(u, v)$

Figure 2: An example of flat traffic limits

$$\begin{aligned} \textit{dest-usage}\,(u, \mathcal{R}) &= \sum_{q \in \mathrm{R},\, u \in \textit{dest}\,(q)} \textit{rate}\,(q) \\ \textit{pp-usage}\,(u, v, \mathcal{R}) &= \sum_{q \in \mathrm{R},\, u \in \textit{src}\,(q),\, v \in \textit{dest}\,(q)} \textit{rate}\,(q) \end{aligned}$$

Let flat traffic limits $\mathcal{T} = (\alpha, \omega, \mu)$ be given, where $\alpha : N \to \mathbb{N}$, $\omega : N \to \mathbb{N}$, and $\mu : N \times N \to \mathbb{N}$. We say that the set of requests $\mathcal{R}$ *is compatible with traffic limits* $\mathcal{T}$ if

$$\begin{aligned} &(\forall u \in N)\ (\textit{src-usage}\,(u, \mathcal{R}) \leq \alpha(u)) \wedge \\ &(\forall u \in N)\ (\textit{dest-usage}\,(u, \mathcal{R}) \leq \omega(u)) \wedge \\ &(\forall u, v \in N)\ (\textit{pp-usage}\,(u, v, \mathcal{R}) \leq \mu(u, v)) \end{aligned} \tag{1}$$

That is, no node is involved in more requests than its termination capacity will allow, and no pair of nodes is involved in more requests than their point-to-point restriction will allow.

There are two restrictions on the values of α, ω, and μ which are sometimes useful. The first is the condition

$$(\forall u, v \in N)(\mu(u, v) \geq \min\{\alpha(u), \omega(v)\}) \tag{2}$$

It can be proven that when this condition holds, the first two lines of condition (1) imply the third line. Therefore the compatible sets of requests depend only upon the values of α and ω. When condition (2) holds, we will call the traffic limits *termination capacity bounded*, or α, ω*-bounded.* The limits of Figure 2 are α, ω-bounded because $\mu(u, v) = \min\{\alpha(u), \omega(v)\}$ for all $u, v \in N$.

The second restriction is

$$\begin{aligned} &(\forall u \in N)\ (\alpha(u) \geq \sum_{v \in N} \mu(u, v)) \wedge \\ &(\forall u \in N)\ (\omega(u) \geq \sum_{v \in N} \mu(v, u)) \end{aligned} \tag{3}$$

When this condition holds, the third line of condition (1) implies the first and second lines. Therefore the compatible sets of requests depend only upon the values of μ. When condition (3) holds, we will call the traffic limits *point-to-point bounded*, or μ*-bounded.*

It is possible for traffic to be neither α, ω-bounded nor μ-bounded. This will be called the *general* case of nonblocking traffic limits.

3.2. A lower bound

In this section we present an algorithm for computing a lower bound on the cost of *any* nonblocking network, using any routing algorithm, given only the traffic limits and the link cost functions. It works when all link cost functions are linear, i.e., $cost_{(u,v)}(x) = \gamma(u,v) \cdot x$ for all pairs u, v. In addition, the coefficients of these linear functions $\gamma(u,v)$ must satisfy the triangle inequality:

$$(\forall u, v, w \in N) \quad \gamma(u,v) \leq \gamma(u,w) + \gamma(w,v)$$

That is, it is never more expensive to build a link directly between a pair of nodes than it is to build links on an indirect path between the nodes.

Since we will compute a lower bound, the method to be described can also be used when each link cost function is at least some linear function. For example, some "staircase" functions, like $40\lceil x/20 \rceil$, are useful as cost functions because they can model the cost of transmission systems where capacity can only be installed in groups of a units ($a = 20$ in this case), and each group costs b units ($b = 40$). This function satisfies $40\lceil x/20 \rceil \geq 2x$. Therefore a lower bound can be computed by the methods in the next sections by assuming that the link cost function is $2x$ for the given link.

A lower bound is useful when we have an instance I of a network design problem and a network design of cost C which is nonblocking, but we do not know how close its cost is to the minimum cost possible. Suppose we have computed a lower bound $L(I)$ on the cost of any nonblocking network for I. In particular, $L(I)$ is a lower bound on the cost of the cheapest solution, $OPT(I)$. Therefore

$$\frac{C}{OPT(I)} \leq \frac{C}{L(I)}$$

For example, if $C = 125$ and $L(I) = 100$, then we know that our solution costs at most 25% more than the minimum cost solution, and it may be closer.

Let the flat traffic limits be given by $\mathcal{T} = (\alpha, \omega, \mu)$. Consider a single point-to-point connection request from u to v with rate r that is compatible with $\mathcal{T}$. In any network that is nonblocking for $\mathcal{T}$, there must be a path from u to v such that all links in the path have at least r units of bandwidth. Given that the link cost functions are linear and the coefficients γ satisfy the triangle inequality, the cheapest network that can be built, to handle this request only, is the one containing the single link (u,v) with capacity r. The cost of this network is $r \cdot \gamma(u,v)$, which is a lower bound on the cost of any network that is nonblocking for $\mathcal{T}$, regardless of the routing algorithm used.

Similarly, for any compatible set of point-to-point connection requests $\mathcal{R}$. The cheapest network that can handle $\mathcal{R}$ is the one which contains links directly between the node pairs involved in requests. The cost of this network is

$$L(I, \mathcal{R}) \quad = \quad \sum_{q \in \mathcal{R}} rate(q) \cdot \gamma(src(q), dest(q))$$

and this quantity is a lower bound on the cost of any network that is nonblocking for $\mathcal{T}$.

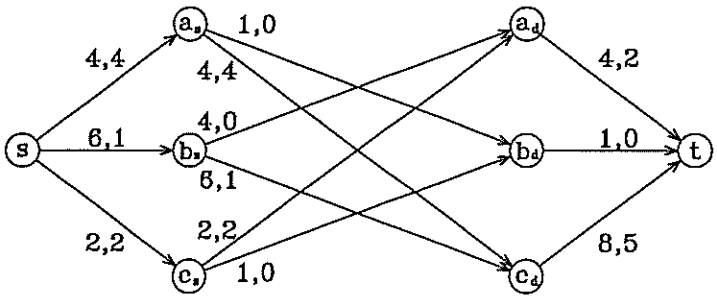

Figure 3: Lower bound network for traffic limits of Figure 2

To find the largest lower bound of this form possible, we wish to compute the value

$$L(I) = \max_{\mathcal{R} \in ALLREQ_{\mathcal{T}}} L(I, \mathcal{R})$$

where $ALLREQ_{\mathcal{T}}$ contains all sets of point-to-point connection requests compatible with $\mathcal{T}$.

Computing $L(I)$ by enumerating all elements of the set $ALLREQ_{\mathcal{T}}$ would be computationally prohibitive. We can compute it much more efficiently by exploiting a relationship between all request sets and all flows in a structure called the lower bound network.

The lower bound network $\mathcal{L} = (V, E, cap)$ is a directed graph with capacities on the edges, $cap : E \to \mathbb{N}$. It is defined below in terms of the network nodes N and the traffic limits $\mathcal{T} = (\alpha, \omega, \mu)$. The edges are given in the form (u, v, c), where the edge is from vertex u to vertex v and has capacity c.

$$\begin{aligned} V &= \{s, t\} \cup \{u_s, u_d : u \in N\} \\ E &= \{(s, u_s, \alpha(u)),\ (u_d, t, \omega(u)) : u \in N\} \cup \\ &\quad \{(u_s, v_d, \mu(u, v)) : u, v \in N, u \neq v\} \end{aligned}$$

The subscript s is short for "source", and the subscript d is short for "destination".

The lower bound network for the flat traffic limits given in Figure 2 is shown in Figure 3. The first number that labels each edge is the capacity. The second number will be explained in the example below.

An important property of this network is that for any compatible set of point-to-point requests $\mathcal{R}$, there is a unique corresponding integer-valued network flow $f_{\mathcal{R}} : E \to \mathbb{N}$ (see any of [FF64, Hu69, Tar83] for definition of a flow) that does not exceed any of the edge capacities. This flow can be constructed by starting with the 0 flow, and then for each request $(u, v, rate) \in \mathcal{R}$, add flow $rate$ along the path $s - u_s - v_d - t$. For example, the set of compatible requests $\mathcal{R} = \{(a, c, 4), (b, c, 1), (c, a, 2)\}$ has the unique corresponding flow $f_{\mathcal{R}}$ which is given by the second numbers labeling the edges in Figure 3.

Note that the flow on edges of the form (s, u_s) is exactly *src-usage*$(u, \mathcal{R})$, flow on edges (u_d, t) is exactly *dest-usage*$(u, \mathcal{R})$, and flow on edges (u_s, v_d) is exactly *pp-usage*$(u, v, \mathcal{R})$. Given this correspondence, it is easily seen that the conditions "$\mathcal{R}$ is compatible" (given

by (1)) and "$f_{\mathcal{R}}$ does not exceed any edge capacities in $\mathcal{L}$" are equivalent. Also note that the value of the flow, $v(f_{\mathcal{R}}) = \sum_{u \in N} f_{\mathcal{R}}(s, u_s)$, is equal to the total rate of all requests in $\mathcal{R}$.

Conversely, for any integer flow f that does not exceed the edge capacities, there is at least one corresponding compatible point-to-point request set $\mathcal{R}_f$. Such a set can be constructed by finding any path $p = s - u_s - v_d - t$ in $\mathcal{L}$ with positive flow, picking any integer quantity $\delta > 0$ that is no larger than the flow on any edge of p, reducing the flow on p by δ, and adding a connection request (u, v, δ) to $\mathcal{R}_f$. Repeat this process until the flow f is 0. It is easy to see that any such $\mathcal{R}_f$ must be compatible. For example, the flow of Figure 3 has several corresponding compatible request sets. In addition to the one given above, there is also $\{(a, c, 3), (a, c, 1), (b, c, 1), (c, a, 1), (c, a, 1)\}$.

Because of the above properties, we say that there is a many to one correspondence from compatible sets of point-to-point request sets to integer flows.

The value of the maximum flow in the lower bound network is just the maximum possible total rate of connections that can exist simultaneously. Instead of the maximum total rate, a simple sum of connection rates, we want the maximum value of $L(I, \mathcal{R})$, which can be thought of as a weighted sum of connection rates, weighted by the $\gamma(\cdot)$ coefficients.

This problem can be modeled as a maximum *cost* flow problem [Tar83, Section 8.4]. An instance of this problem is a directed graph with edge costs as well as edge capacities. The cost of an edge represents the cost per unit of flow on that edge. The cost of a flow is the sum over all edges of the flow on the edge multiplied by the edge cost. For the lower bound network, define the costs of edges (s, u_s) and (v_d, t) to be 0, and for each pair u, v, define the cost of edge (u_s, v_d) to be $\gamma(u, v)$.

Now, given any compatible set of point-to-point requests $\mathcal{R}$, the cost of its corresponding integer flow f will be exactly $L(I, \mathcal{R})$. The problem of determining the most costly set of requests to satisfy has become a maximum cost flow problem in the lower bound network.

In the literature, this problem is often called the minimum cost flow problem or the minimum cost circulation problem. Simply negate the edge costs given above to convert it to a minimum cost flow problem. The most efficient algorithms for this problem known to the author are cited by Goldberg and Tarjan [GT90].

We close this section with an example that shows why we restrict the cost coefficients $\gamma(\cdot)$ to those that satisfy the triangle inequality. The example does not satisfy the triangle inequality, and the lower bound computed by the method above is larger than the cost of a nonblocking network, and hence it is not truly a lower bound at all.

The example instance has three nodes a, b, and c. All α, ω, and μ values equal 1, and the cost coefficients are $\gamma(a, b) = \gamma(b, a) = 1$, $\gamma(b, c) = \gamma(c, b) = 2$, and $\gamma(a, c) = \gamma(c, a) = 4$. These costs violate the triangle inequality because $\gamma(a, c) > \gamma(a, b) + \gamma(b, c)$. The lower bound has value 8, as shown by the most costly request set $\{(a, c, 1), (c, a, 1)\}$. A network that is nonblocking for these traffic limits contains the links (a, b), (b, a), (b, c), and (c, b), all with capacity 1. The network has cost 6.

This occurs because the lower bound is made high by using the large cost $\gamma(a, c) = 4$ directly between a and c, but the nonblocking network can get between a and c more

cheaply by going through b. This cannot happen when the $\gamma(\cdot)$ coefficients satisfy the triangle inequality.

3.3. Link dimensioning

In this section we describe an algorithm for computing the minimum necessary link capacities of a nonblocking network, given the traffic limits and the fixed path table to be used. An important property of fixed path routing is that this computation can be done independently for each link. We specify the problem as one of minimizing the cost of a nonblocking network.

Nonblocking network configuration with fixed paths given

INSTANCE: A set of nodes N. For each node pair u, v, a nondecreasing function $cost_{u,v} : \mathbb{N} \to \mathbb{N}$, where $cost_{u,v}(x)$ is the cost of installing a link of capacity x from u to v. Traffic limits $\mathcal{T}$ are given by (α, ω, μ). For each $u, v \in N$, a directed path $path(u, v)$ from u to v.

SOLUTION: A capacity $cap(u, v) \in \mathbb{N}$ for each node pair u, v. This assignment of capacity should make the network nonblocking for traffic $\mathcal{T}$ and the fixed path routing algorithm with fixed paths *path*, where connection requests may be multirate and multipoint.

SOLUTION COST: The cost of the network is the sum of the costs of each link:

$$\sum_{u,v \in N} cost_{u,v}(cap(u, v))$$

OBJECT: Find a solution with minimum cost.

In Section 3.3.1, we present some properties of the fixed path routing algorithm which hold for any table *path*, and any traffic limits, including the hierarchical traffic limits studied later. Section 3.3.2 shows how to solve the problem above for tree-shaped networks and when requests and connections must be point-to-point. Later we will extend this to include non-tree networks, and show that all link-dimensioning algorithms also work when multipoint connections are allowed.

3.3.1. Properties of fixed path routing. The first fact to note about fixed path routing is: no matter what sequence of requests to add and drop connections came before, *the state of the network is a function of the current set of requests.*

Assume for the moment that all links which are used in some fixed path have a very large capacity which could not be exceeded even if every connection used it, e.g., $\min\{\alpha(N), \omega(N)\}$. Let h be the function that maps request sets to states (this function depends only upon the table of fixed paths *path*). Given any set of point-to-point requests $\mathcal{R}$, we can determine the state of the network $h(\mathcal{R})$, and therefore the usage of each link l, $usage(l, h(\mathcal{R}))$. The set of all point-to-point request sets $ALLREQ_{\mathcal{T}}$ which are compatible with the given traffic limits

$\mathcal{T}$, $ALLREQ_{\mathcal{T}} = \{\mathcal{R} : \mathcal{R}$ is point-to-point and compatible with $\mathcal{T}\}$, is finite. Therefore we can determine the maximum possible usage of any given link l:

$$maxusage(l) = \max_{\mathcal{R} \in ALLREQ_{\mathcal{T}}} usage\,(l, h(\mathcal{R}))$$

Note that this value is independent of any other link capacity. It depends only upon *path*, $\mathcal{T}$, and l.

If we assign any link capacities *cap* that satisfy $cap\,(l) \geq maxusage(l)$ for all links l, then the network will be nonblocking. If any link l has capacity less than $maxusage(l)$, we can cause the network to block by making the requests in a set $\mathcal{R}^*$ in any order, where $usage\,(l, h(\mathcal{R}^*)) = maxusage(l)$. Therefore, when the link cost functions $cost_{u,v}$ are *any* nondecreasing functions of capacity, the link capacities $cap\,(l) = maxusage(l)$, for all $l = (u, v)$, $u, v \in N$, will be the cheapest solution to the computational problem above. This is true for any traffic limits and fixed paths. Therefore the computational problem above has been reduced to computing the function *maxusage*.

Let $l \in L$ be any link of the network. Computing $maxusage(l)$ by generating every compatible request set in $ALLREQ_{\mathcal{T}}$ would be computationally prohibitive. It may be computed more efficiently by exploiting the properties of the lower bound network defined in section 3.2, and some modifications of it.

We begin transforming the expression for *maxusage* by noting that $usage\,(l, h(\mathcal{R}))$ is equal to the total rate of those requests in $\mathcal{R}$ whose routes must use link l. Define $\mathcal{R}_l$ to be this set, for any $\mathcal{R}$ and l. That is, $\mathcal{R}_l = \{q \in \mathcal{R} : u \in src\,(q), v \in dest\,(q), l \in path(u, v)\}$. Then we may write $usage\,(l, h(\mathcal{R})) = \sum_{q \in \mathcal{R}_l} rate\,(q)$, and therefore

$$maxusage(l) = \max_{\mathcal{R} \in ALLREQ_{\mathcal{T}}} \left(\sum_{q \in \mathcal{R}_l} rate\,(q) \right) \tag{4}$$

3.3.2. Link dimensioning in tree-shaped networks. In this section we restrict the union of the path links in the table *path* to be a tree-shaped network. A tree-shaped network is a directed graph obtained by starting with a tree (i.e., a connected, acyclic, undirected graph) and replacing each undirected edge with two oppositely directed edges between the same pair of vertices. For example, the graph of Figure 4 is a tree-shaped network.

For any link l in a tree-shaped network $\mathcal{N}$, let X_l be the set of nodes on its "source side". That is, if $l = (u, v)$, then X_l is the set of nodes in the connected component of $\mathcal{N} - \{(u, v), (v, u)\}$ which contains node u. All other nodes, those in $N - X_l$, are on the "destination side" of l. For example, the link (b, c) in the network of Figure 4 has the nodes $X_{\{b,c\}} = \{a, b, d, e\}$ on its source side and $\{c, f, g, h\}$ on its destination side.

In a tree-shaped network, all node pairs which use any given link l are of the form u, v, where $u \in X_l$ and $v \in N - X_l$.

Consider the lower bound network $\mathcal{L}$. Since nodes in X_l will never receive connections which use link l, we may model this by reducing their ω values to 0. That is, reduce the

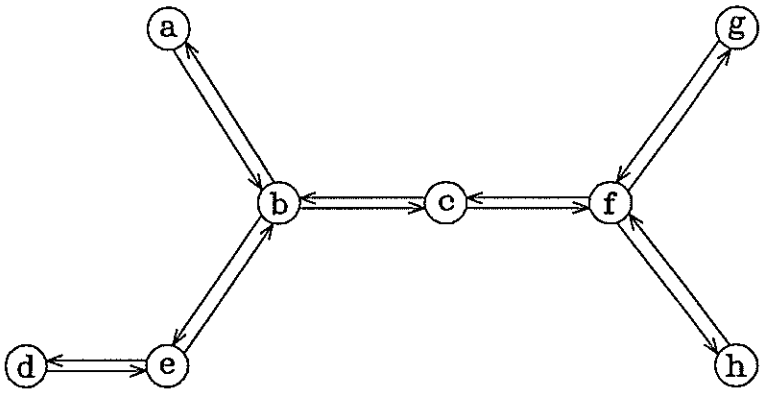

Figure 4: A tree-shaped network

capacities of edges of the form (u_d, t), where $u \in X_l$, to 0 (or, equivalently, remove them). Similarly, nodes in $N - X_l$ will never initiate connections which use link l, so reduce their α values to 0 by reducing the capacities of the edges (s, v_s), where $v \in N - X_l$, to 0. Call this modified lower bound network $\mathcal{L}_l$.

Just as there is a many to one correspondence from compatible sets of point-to-point requests $\mathcal{R}$ to integer flows in $\mathcal{L}$, there is a many to one correspondence from sets $\mathcal{R}_l$ to integer flows in $\mathcal{L}_l$, where $\mathcal{R}$ is a compatible set of point-to-point requests. The value of such a flow f, $v(f)$, is equal to the total rate of requests in any corresponding set $\mathcal{R}_l$. Therefore we see that

$$\begin{aligned} maxusage(l) &= \max_{\mathcal{R} \in ALLREQ_{\mathcal{T}}} \left(\sum_{q \in \mathcal{R}_l} rate(q) \right) \\ &= \max_{f \in ALLFLOWS_{\mathcal{L}_l}} v(f) \end{aligned} \tag{5}$$

where $ALLFLOWS_{\mathcal{L}_l}$ is the set of all integer flows in the network $\mathcal{L}_l$. The first line is just Equation (4), and (5) follows from the many to one correspondence from request sets $\mathcal{R}$ to flows f.

Determining the flow with maximum value is simply a maximum flow problem, for which many efficient algorithms have been designed. King, Rao, and Tarjan [KRT92] have designed an algorithm with a worst-case running time of $O(mn + n^{2+\epsilon})$ for any $\epsilon > 0$, where m is the number of edges in the network, and n is the number of vertices. This algorithm has the best asymptotic efficiency known to the author. Goldberg and Tarjan [GT88, Section 4] describe an algorithm which is quite easy to implement with a worst-case running time of $O(n^3)$.

In the special case when the traffic is α, ω-bounded, it is easy to see that the minimum cut in $\mathcal{L}_l$ is either $(\{s\}, V - \{s\})$, with value $\alpha(X_l)$, or $(V - \{t\}, \{t\})$, with value $\omega(N - X_l)$. By the max-flow, min-cut theorem [FF64], the value of the maximum flow is then equal to $\min\{\alpha(X_l), \omega(N - X_l)\}$. This special case will be used often later.

3.4. Experimental results

In this section, we compare the value of the lower bound to the costs of easily computable nonblocking networks. This is done for randomly generated problem instances.

The easily computable nonblocking networks are star networks, in which there is a "center" node c, and all other nodes are directly attached to c by two oppositely directed links. There are only $n = |N|$ such networks, and we can compute the cost of each one quickly by using the link dimensioning algorithm of the previous section.

A simple implementation of this method would require computing $2(n-1)$ maximum flows (one for each link) for each of the n possible centers. We can improve on this by computing only $2n$ maximum flows total. For any node u, we can compute the capacities of the links in and out of u, provided that u is not the center node. These capacities are the same no matter which node is the center, and so may be computed once.

The restricted form of each maximum flow instance allows them to be computed in $O(n)$ time each. After computing all of these capacities, the cost of each possible star network can also be computed in $O(n)$ time each, giving a total running time of $O(n^2)$ to find the cheapest star network.

In the following, whenever we say that a value is generated randomly in some interval, we mean that it is generated from a uniform distribution on the interval. Similarly, when we place a point randomly in some rectangle, we mean that its location is generated randomly with a uniform distribution on the area.

A single experiment consists of choosing a number of nodes n, a range of termination capacity values $[\alpha_{\text{lo}}, \alpha_{\text{hi}}]$, and a range of real numbers $[\mu_{lo}, \mu_{hi}]$, which is a sub-interval of $[0, 1]$. Generate a random flat instance as follows. All nodes are placed randomly in a unit square. Link cost coefficients $\gamma(u, v)$ are set equal to the Euclidean distance between u and v. For each node u, choose the integer $\alpha(u)$ randomly in the interval $[\alpha_{\text{lo}}, \alpha_{\text{hi}}]$, and set $\omega(u) = \alpha(u)$. For each node pair u, v, choose a real value x randomly in the interval $[\mu_{lo}, \mu_{hi}]$ and set $\mu(u, v) = x \cdot \min\{\alpha(u), \omega(v)\}$. Note that if $\mu_{lo} = \mu_{hi} = 1$, then the traffic limits α, ω, μ will always be α, ω-bounded.

After the instance has been generated, the cost of the cheapest star network was computed as described above, and the lower bound is computed by the algorithm in Section 3.2. The *performance ratio*, which is a real value no less than 1, is equal to the cost of the star network divided by the lower bound.

In Figure 5, each data point is the average of the performance ratios of 50 randomly generated instances, all generated with the same values of n, $\alpha_{\text{lo}} = 10$, $\alpha_{\text{hi}} = 20$, and $\mu_{lo} = \mu_{hi} = 1$. Experiments were done for values of n ranging over the set $\{3, 4, 5, 6, \ldots, 14, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100\}$. The same experiments were performed with α randomly drawn from the interval $[1, 30]$, and with all α values equal to 10. The resulting plots are not significantly different than those in Figure 5, so they have not been shown.

Note that even at the worst (highest) part of the curve for small n, the average performance ratio is no more than about 1.08. This shows that the star network solutions are within 8% of optimal on average for small n, and even closer for large n. The maximum

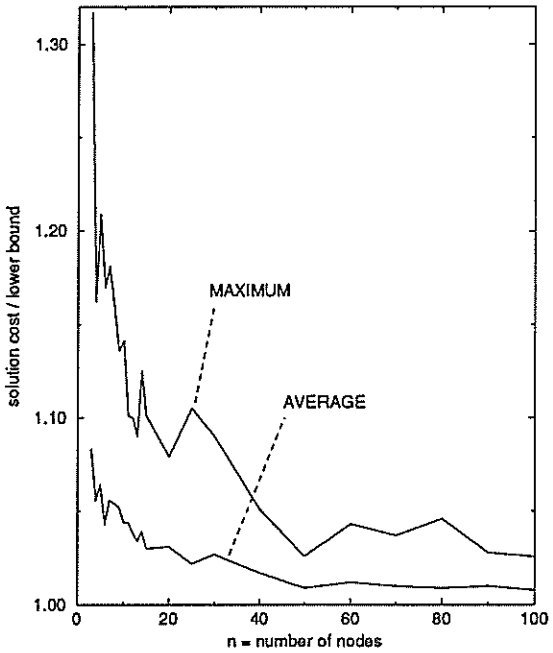

Figure 5: Experimental results for flat α, ω-bounded traffic

(not the average) of all 50 of the individual performance ratios for $n = 3$ is 1.317. For $n \leq 7$, we have exhaustively enumerated all n^{n-2} tree-shaped networks of the nodes, not just star networks, to find the one that gives the cheapest nonblocking network. In every case, a star network was among the cheapest solutions. This prompted the search for a proof that the minimum cost star network is the cheapest among all tree-shaped networks. The proof was found and is presented in Section 3.5.

When the performance ratio is large, it means that either the lower bound is far below the optimal cost, the cheapest star network is far above the optimal cost, or both. We conjecture that the lower bound is far below the optimal cost for small n, and the minimum cost star network is exactly optimal. More precisely, we conjecture that for α, ω-bounded traffic with $\alpha(u) = \omega(u)$ for all u, link cost functions which are linear, and link cost coefficients satisfying the triangle inequality, the cheapest star network is the minimum cost nonblocking network among all nonblocking networks, with any routing algorithm. The proof mentioned above proves that the minimum cost star is cheapest among all tree networks, but not necessarily among all solutions.

When n gets large, we see that the performance ratio gets closer to 1. This means that both the lower bound and the minimum cost star network are getting closer to the optimal value. This prompted the search for a proof that the curve does approach 1 as n gets large. In Section 3.6, we show that the probability the performance ratio is at most $1 + \epsilon$ goes to 1 as n gets large, for any $\epsilon > 0$. This result holds for random instances with uniformly distributed nodes, and α, ω-bounded traffic limits in which each α and ω value can be generated with any distribution desired, even with different distributions for different nodes, as long as the averages of all distributions are the same, and it is impossible for any distribution to generate a value outside of the interval $[\alpha_{\text{lo}}, \alpha_{\text{hi}}]$. This proof should not be too difficult to extend to more general kinds of traffic limits.

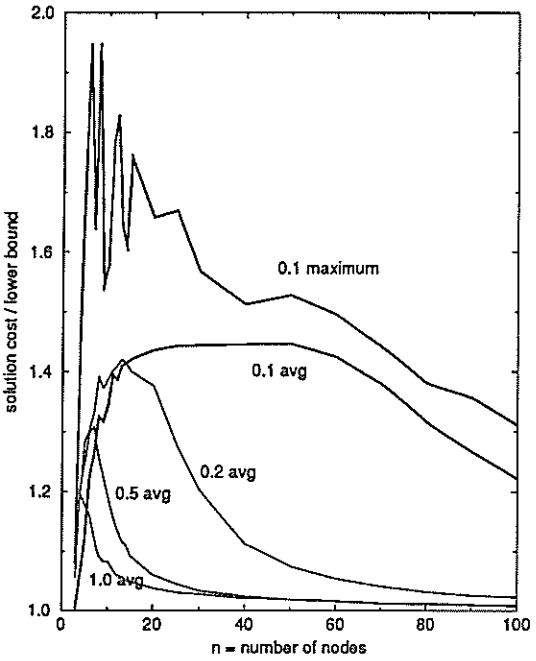

Figure 6: Experimental results for flat general traffic

Figure 6 shows several curves. All instances generated used the parameters $\alpha_{\text{lo}} = 10$, $\alpha_{\text{hi}} = 20$ as before, but now we set $\mu_{lo} = 0$ and each curve represents a different value of μ_{hi}, as labeled. These are general traffic limits, i.e., neither α, ω-bounded nor μ-bounded, although they may happen to be μ-bounded for small n and small values of μ_{hi}. 50 random instances were generated for each data point, and the same set of values of n was used as before.

The performance ratios are much worse on averge now. For smaller values of μ_{hi}, the performance ratios stay high for a longer time. This can be explained by examining the kinds of maximum cost flows which we get in the lower bound network, and the capacities of star network links.

Suppose that $\mu(u, v)$ is equal to its maximum possible value, $U = \min\{\alpha(u), \omega(v)\}$. Then up to U units of flow are free to go from u_s to v_d in the lower bound network. If all $\mu(u, v)$ values are at their maximum value, then flow is free to travel between pairs of nodes which are furthest apart, making the lower bound value large. When $\mu_{hi} = 0.1$, however, then the expected value of $\mu(u, v)$ is $0.1/2 = 1/20$ of U, and now flow from u must be split among 20 other nodes, on average. When there are few nodes, not many of them will be far away, and so flow must be sent to close nodes, yielding a smaller lower bound. As n grows, it is more likely that there will be 20 nodes which are far away, increasing the lower bound.

A similar examination of the capacities required on the star network links show that one would not expect them to be much smaller than when μ values are as large as possible. Therefore, the minimum star network costs do not decrease as much as the lower bound does, and the performance ratio increases.

However, all curves eventually start to approach 1. This gives empirical evidence that we should be able to extend the probabilistic result of Section 3.6 to more general kinds of

flat traffic limits.

When μ values are small, the traffic limits are more likely to be "close" to being μ-bounded. The closer the traffic is to satisfying this condition, then the more likely it is that a complete network, with direct links between every node pair, is cheaper than any tree solution. When the traffic limits are μ-bounded, the complete network is the optimal solution [Fin92].

All of the curves except the top one are average performance ratios, averaged over 50 instances. The top curve is the maximum performance ratio among all 50 instances. Similar curves for the other values of μ_{hi} would only clutter the plot. They are typically about twice as high above 1 as the corresponding average curve. The worst performance ratio over all was 2.051 for a four node instance with $\mu_{hi} = 0.2$.

3.5. A star network is always cheapest among all trees

Define $\alpha(X) = \sum_{u \in X} \alpha(u)$ for any set X, $X \subseteq N$. Define $\omega(X)$ similarly.

THEOREM 3.1. *Let flat α, ω-bounded traffic limits α, ω be given for a set of nodes N, where $\alpha(N) = \omega(N)$, and let all link cost functions be linear with coefficients $\gamma(u, v)$ satisfying the triangle inequality. Then a star network is cheapest among all nonblocking tree-shaped networks.*

We prove this theorem by showing that for any tree-shaped network, there is always a "center" node c with the following property. If any node of the tree is not adjacent to c, then we may cut it from its current place in the tree, make it adjacent to c, and the resulting tree-shaped network has nonblocking link capacities which are no more expensive than the original. By repeating this process at most $n - 2$ times, the result is eventually a star network with center node c which is no more expensive than the original network. Hence, for every tree-shaped network which is not a star, there exists a star which is no more expensive.

Note that this does not say that the star with center c will always have minimum cost among all stars. However, the theorem shows us that we may find the minimum cost tree network by trying all n stars exhaustively.

Let $T = (V, E)$ be an undirected tree where each vertex v has a real weight $w(v) \geq 0$. Define $w(X)$ to be the sum of the weights of vertices in X, where X is either a subset of V or a subgraph of T. For any two distinct nodes u, v in T, let $f(u, v)$ be the first edge on the unique path from u to v in T, and define the *subtree of v with respect to u*, $S_u(v)$, to be the connected component of $T - f(v, u)$ which contains v. A vertex $c \in V$ is called a *weighted tree center* if $w(S_c(v)) \leq w(T)/2$ for all vertices v adjacent to c.

LEMMA 3.1. *Let $T = (V, E)$ be an undirected tree, where each node $v \in V$ has a nonnegative weight $w(v)$. Then there exists a weighted tree center $c \in V$.*

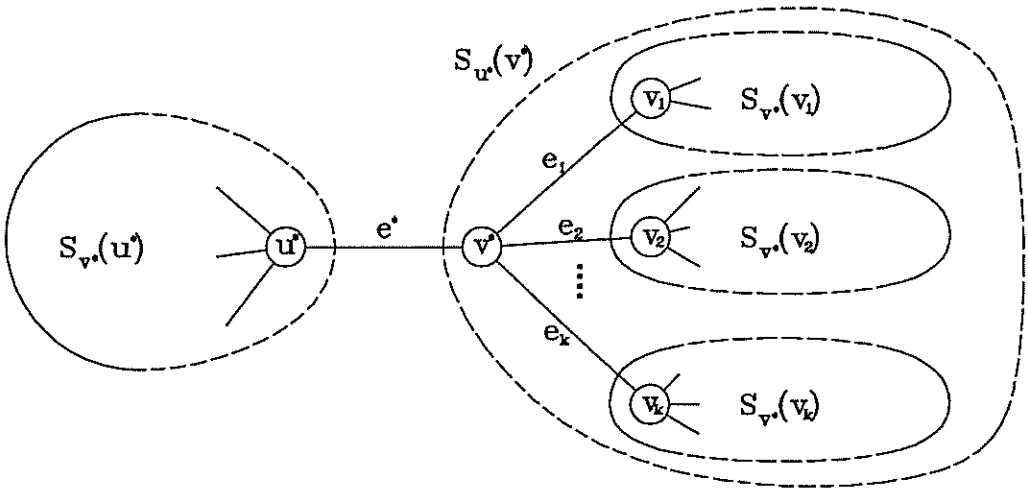

Figure 7: Tree structure in proof of Lemma 3.1

Proof: For any edge $e \in E$, define the *imbalance* of e, $I(e)$, to be $|w(S_u(v)) - w(S_v(u))|$, where $e = \{u, v\}$.

Let $e^* = \{u^*, v^*\}$ be an edge with minimum imbalance. Note that

$$(\forall u, v)\ \ \{u, v\} \in E \ \Rightarrow\ w(S_u(v)) + w(S_v(u)) = w(T) \tag{6}$$

If $I(e^*) = 0$, then $w(S_{u^*}(v^*)) = w(S_{v^*}(u^*)) = w(T)/2$, and both u^* and v^* are weighted tree centers.

If $I(e^*) > 0$, then suppose without loss of generality that $w(S_{v^*}(u^*)) < w(S_{u^*}(v^*))$, and so

$$I(e^*) = w(S_{u^*}(v^*)) - w(S_{v^*}(u^*)) \tag{7}$$

From Equation (6) it also follows that $w(S_{v^*}(u^*)) < w(T)/2 < w(S_{u^*}(v^*))$.

If u^* is the only vertex adjacent to v^*, then v^* is the only weighted tree center of T. Call this Case 1.

Otherwise, let $v_1, \ldots, v_k$ be the neighbors of v^* other than u^*, and let $e_i = \{v^*, v_i\}$ for all i, $1 \le i \le k$. See Figure 7 for a diagram of the tree structure. Then we have the following for all i:

$$\begin{aligned} I(e_i) &= |w(S_{v_i}(v^*)) - w(S_{v^*}(v_i))| && \{\text{ Defn. of } I\ \} \\ &= |w(T) - 2w(S_{v^*}(v_i))| && \{\text{ Equation (6) }\} \end{aligned}$$

By the choice of e^*, we know that $(\forall i) I(e_i) \ge I(e^*)$. $|x| \ge y$ if and only if ($x \ge y$ or $-x \ge y$). Therefore $(\forall i) I(e_i) \ge I(e^*)$ is true if and only if

$$(\forall i)\ \ (w(T) - 2w(S_{v_*}(v_i)) \ge I(e^*)) \text{ or } \ (2w(S_{v_*}(v_i)) - w(T) \ge I(e^*))$$

Using Equations (6) and (7) we may derive the equivalent condition

$$(\forall i)\ \ w(S_{v^*}(v_i)) \le w(S_{v^*}(u^*)) \text{ or } \ w(S_{v^*}(v_i)) \ge w(S_{u^*}(v^*))$$

There are now two cases two consider. Case 2a is when $(\forall i) w(S_{v^*}(v_i)) \leq w(S_{v^*}(u^*))$. Recall that $w(S_{v^*}(u^*)) < w(T)/2$. Therefore vertex v^* is a weighted tree center.

Case 2b is when there exists a vertex v_i such that $w(S_{v^*}(v_i)) \geq w(S_{u^*}(v^*))$. Since $S_{v^*}(v_i)$ is a subgraph of $S_{u^*}(v^*)$ and all node weights are nonnegative, we must have $w(S_{v^*}(v_i)) = w(S_{u^*}(v^*))$. This implies that $w(z) = 0$ for all vertices z in $S_{u^*}(v^*) - S_{v^*}(v_i)$. Now edge $\{v^*, v_i\}$ is also an edge of minimum imbalance, and we can "repeat the proof" on this edge. Since the graph is finite, this repetition will eventually halt with either Case 1 or Case 2a. ■

For any undirected tree $T = (V, E)$, define $\mathcal{N}(T)$ to be the tree-shaped nonblocking network which is obtained from T by replacing each undirected edge $\{u, v\}$ with the directed links (u, v) and (v, u). The capacities of links in $\mathcal{N}(T)$ are the minimum necessary to make the network nonblocking. Recall from section 3.3.2 that for α, ω-bounded traffic, the minimum necessary capacity of a tree link with nodes X on its source side and $N - X$ on its destination side is $\min\{\alpha(X), \omega(N - X)\}$.

LEMMA 3.2. *Let flat α, ω-bounded traffic limits α, ω be given for a set of nodes V, where $\alpha(V) = \omega(V)$, and let all link cost functions be linear with coefficients $\gamma(u, v)$ which satisfy the triangle inequality. Let $T = (V, E)$ be an undirected tree, and let c be a weighted tree center for T where $w(v) = \alpha(v) + \omega(v)$ for all $v \in V$. For any $v \neq c$, the network $\mathcal{N}(T - f(v, c) + \{v, c\})$ costs no more than $\mathcal{N}(T)$.*

Proof: If v is adjacent to c, then $T - f(v, c) + \{v, c\} = T$, so the network does not change.

If v is not adjacent to c, then let the unique path from c to v in T be $p = u_0 u_1 \ldots u_k$, where $k \geq 2$, $c = u_0$, and $v = u_k$. Define

$$T_i = \begin{cases} S_{u_1}(c) & \text{if } i = 0 \\ S_c(u_i) - S_c(u_{i+1}) & \text{if } 1 \leq i \leq k-1 \\ S_c(u_k) & \text{if } i = k \end{cases}$$

See Figure 8 for an annotated picture of $\mathcal{N}(T)$ and its parts.

The network $\mathcal{N}(T - f(v, c) + \{v, c\})$ is depicted in Figure 9. Note that for any link which is not on the path from v to c, its minimum necessary capacity is the same in $\mathcal{N}(T)$ and $\mathcal{N}(T - f(v, c) + \{v, c\})$ (this is true even for hierarchical traffic limits). The only difference between the networks is that the capacities of links on the path p may change, and the links (c, v) and (v, c) are added.

The minimum necessary capacity of the new link (c, v) is $\min\{\alpha(T - T_k), \omega(T_k)\}$. Derivations below will show that $\omega(T_k)$ is at most $\alpha(T - T_k)$. Similarly, the capacity of (v, c) is $\alpha(T_k)$. This adds $\omega(T_k)\gamma(c, v) + \alpha(T_k)\gamma(v, c)$ to the cost of the network. Suppose we could show that the capacity of every link (u_i, u_{i+1}) is reduced by $\omega(T_k)$ in the new tree, and the capacity of every link (u_{i+1}, u_i) is reduced by $\alpha(T_k)$ in the new tree. Then the cost of the new tree minus the cost of the old tree is

$$\omega(T_k)\left(\gamma(c, v) - \sum_{j=0}^{k-1} \gamma(u_j, u_{j+1})\right) + \alpha(T_k)\left(\gamma(v, c) - \sum_{j=0}^{k-1} \gamma(u_{j+1}, u_j)\right)$$

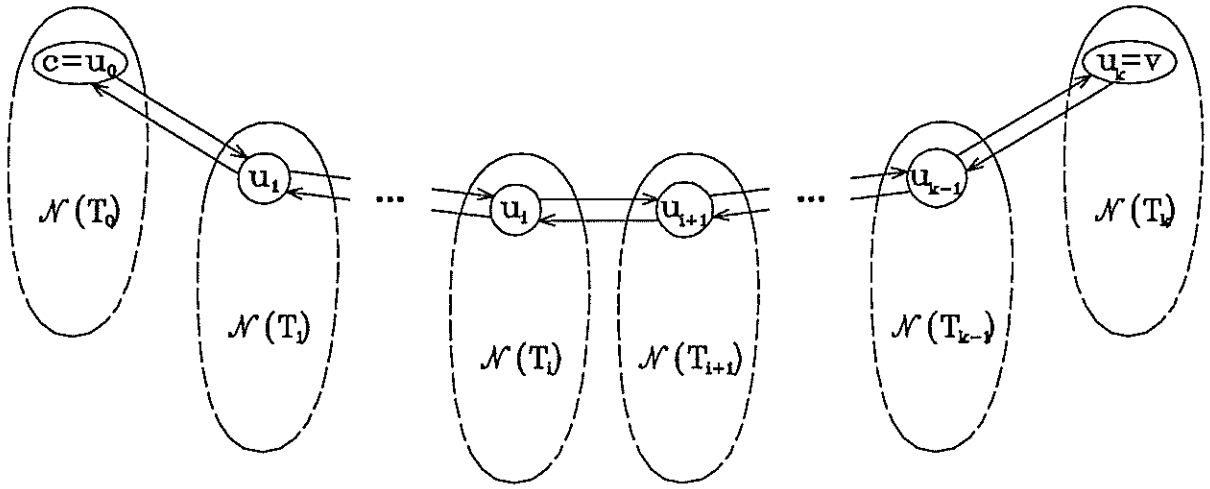

Figure 8: Structure of $\mathcal{N}(T)$ in Lemma 3.2

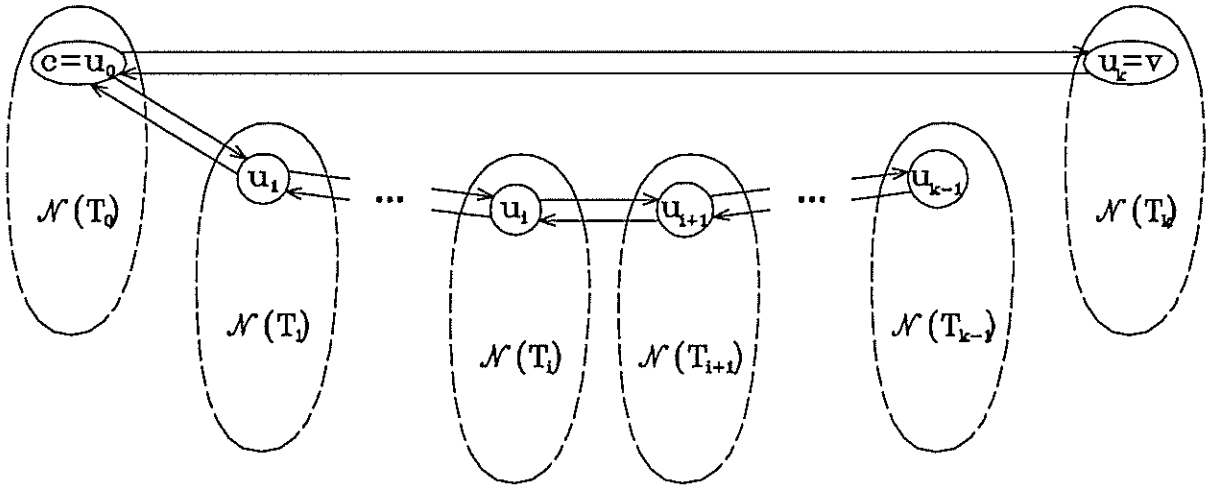

Figure 9: Structure of $\mathcal{N}(T - f(v,c) + \{v,c\})$ in Lemma 3.2

Because the coefficients γ satisfy the triangle inequality, each of the parenthesized subexpressions above is at most 0. Therefore the cost of the new tree is at most the cost of the old tree.

The rest of the proof shows that the capacities of links on the path p do decrease by the amounts given above. We will use *cap* to denote the minimum necessary capacities of links in $\mathcal{N}(T)$, the "before" network, and *cap'* to denote the minimum necessary capacities of links in $\mathcal{N}(T - f(v,c) + \{v,c\})$, the "after" network.

First we compute the capacities of links in $\mathcal{N}(T)$. For all i, $0 \leq i \leq k-1$, the capacity of (u_i, u_{i+1}) must be at least

$$cap\,(u_i, u_{i+1}) = \min\left\{\sum_{j=0}^{i} \alpha(T_j),\ \sum_{j=i+1}^{k} \omega(T_j)\right\} \tag{8}$$

This expression may be simplified by using additional properties of T. Since c is a weighted tree center and u_1 is adjacent to c, we know that $w(S_c(u_1)) \leq w(V)/2$. Using the definition of w and the restriction $\alpha(V) = \omega(V)$ we may conclude

$$\begin{aligned}\sum_{j=1}^{k}(\alpha(T_j) + \omega(T_j)) &\leq (\alpha(V) + \omega(V))/2 \\ &= \alpha(V) = \omega(V)\end{aligned}$$

and by subtracting either $\sum_{j=1}^{k} \alpha(T_j)$ or $\sum_{j=1}^{k} \omega(T_j)$ from each side of the inequality above we get

$$\sum_{j=1}^{k} \omega(T_j) \leq \alpha(V) - \sum_{j=1}^{k} \alpha(T_j) = \alpha(T_0) \tag{9}$$

$$\sum_{j=1}^{k} \alpha(T_j) \leq \omega(V) - \sum_{j=1}^{k} \omega(T_j) = \omega(T_0) \tag{10}$$

Now, for all $0 \leq i \leq k-1$ we may conclude

$$\begin{array}{lll} \sum_{j=i+1}^{k} \omega(T_j) & \leq \sum_{j=1}^{k} \omega(T_j) & \{\sum_{j=1}^{i} \omega(T_j) \geq 0\} \\ & \leq \alpha(T_0) & \{\text{ Inequality (9) }\} \\ & \leq \sum_{j=0}^{i} \alpha(T_j) & \{\sum_{j=1}^{i} \alpha(T_j) \geq 0\} \end{array}$$

That is, the second term of the minimization in Equation (8) is always no larger than the first term, and so the equation may always be written as

$$cap\,(u_i, u_{i+1}) = \sum_{j=i+1}^{k} \omega(T_j) \tag{11}$$

By interchanging α's and ω's in the derivation above, we may also conclude that the minimum necessary capacity of link (u_{i+1}, u_i) is

$$cap\,(u_{i+1}, u_i) = \sum_{j=i+1}^{k} \alpha(T_j) \tag{12}$$

Now we compute the capacities of links in $\mathcal{N}(T - f(v,c) + \{v,c\})$. For all i, $0 \leq i \leq k-1$, the capacity of (u_i, u_{i+1}) must be at least

$$cap'(u_i, u_{i+1}) = \min\left\{\sum_{j=0}^{i} \alpha(T_j) + \alpha(T_k),\ \sum_{j=i+1}^{k-1} \omega(T_j)\right\} \tag{13}$$

Comparing this to Equation (8), we see that the first term in the minimization of (13) is no smaller than the first term of (8), and the second term of (13) is no larger than the second term of (8). Therefore, for the same reason Equation (8) may be written as (11), (13) may be written

$$cap'(u_i, u_{i+1}) = \sum_{j=i+1}^{k-1} \omega(T_j) \tag{14}$$

Again, by interchanging α's and ω's, we may determine that the minimum necessary capacity of link (u_{i+1}, u_i) is

$$cap'(u_{i+1}, u_i) = \sum_{j=i+1}^{k-1} \alpha(T_j) \tag{15}$$

It is now easy to see that

$$\begin{aligned} cap\,(u_i, u_{i+1}) - cap'(u_i, u_{i+1}) &= \sum_{j=i+1}^{k} \omega(T_j) - \sum_{j=i+1}^{k-1} \omega(T_j) \\ &= \omega(T_k) \end{aligned}$$

and similarly $cap\,(u_{i+1}, u_i) - cap'(u_{i+1}, u_i) = \alpha(T_k)$, as claimed. ∎

Theorem 3.1 does not always hold for instances in which $\alpha(V) \neq \omega(V)$. For example, consider the class of instances which have n nodes placed at distinct points on a straight line, where each consecutive pair of nodes is 1 unit apart. Let the link cost coefficients $\gamma(u, v)$ equal the distance between the nodes. Number the nodes in the order they appear on the line from 1 to n. Let $\alpha(u) = 1$ and $\omega(u) = n - 1$ for all $u \in V$.

A cheap solution is the tree, but not star, which has links between consecutive pairs of nodes. To make this network nonblocking, the capacity of link $(i, i+1)$ must be i, and the capacity of link $(i+1, i)$ must be $n - i$. The total cost of this network is $n(n-1)$.

Any star network for this instance will have nonblocking link capacities such that links directed into the center have capacity 1 and links directed away from the center have capacity $n - 1$. We can easily show that the cost of the star with center node i is

$$\frac{n}{2}(i(i-1) + (n-i)(n-i+1))$$

which has its minimum value of $n(n-1)(n+1)/4$ when $i = (n+1)/2$. This is larger than $n(n-1)$ for all $n \geq 4$.

Therefore it is not possible to generalize the theorem to arbitrary values of $\alpha(V)$ and $\omega(V)$. This example has $\omega(V) = (n-1)\alpha(V)$, so it may be possible to generalize the theorem to cases when $\alpha(V)$ and $\omega(V)$ are "close enough", but not equal.

3.6. A probabilistic result

Suppose that flat α,ω-bounded instances I are randomly generated by the method given in Section 3.4, except that now the locations are randomly chosen in the interior of a unit circle on the plane, i.e., a unit disk. Also, $\alpha(u)$ and $\omega(u)$ values may be chosen randomly with any distribution desired, with the restrictions that the expected values of all such distributions equal $\overline{\alpha}$, and it is impossible to generate values outside of the interval $[\alpha_{\rm lo}, \alpha_{\rm hi}]$.

Let $A(I)$ be the cost of the cheapest star network for instance I, and $L(I)$ be the value of the lower bound for instance I.

THEOREM 3.2. *Let $\delta > 0$ be given. When instances of the nonblocking network design problem are generated as described above, then*

$$\lim_{n\to\infty} \Pr\left\{\frac{A(I)}{L(I)} \le 1+\delta\right\} = 1$$

This theorem can also be proven when node locations are uniformly distributed in the unit square, or many shapes which have their areas symmetrically arranged around their center points. Using the unit disk just makes some mathematical expressions in the proof simpler.

The proof is done in two parts. Lemma 3.3 states that a randomly generated instance I is "balanced" with probability approaching 1 as n goes to infinity. Lemma 3.6 then shows that $A(I)/L(I)$ is close to 1 for all balanced instances I.

For the first lemma, we will need a theorem of Hoeffding [Hoe63] and another due to Angluin and Valiant [AV79, HR90].

THEOREM 3.3. *(Hoeffding) Let X_i, $1 \le i \le n$, be independent random variables, each having mean μ and $\Pr\{a \le X_i \le b\} = 1$. Then for any real t, $0 < t < b-\mu$, we have*

$$\begin{aligned}
\Pr\left\{\sum_{i=1}^{n} X_i \ge n\mu + nt\right\} &\le e^{-2nt^2/(b-a)^2} \\
\Pr\left\{\sum_{i=1}^{n} X_i \le n\mu - nt\right\} &\le e^{-2nt^2/(b-a)^2}
\end{aligned}$$

THEOREM 3.4. *(Angluin,Valiant) Let X_i, $1 \le i \le n$, be independent random variables, each of which has probability p of being 1 and probability $1-p$ of being 0. Then for any t, $0 \le t \le 1$*

$$\begin{aligned}
\Pr\left\{\sum_{i=1}^{n} X_i \ge (1+t)np\right\} &\le e^{-t^2np/3} \\
\Pr\left\{\sum_{i=1}^{n} X_i \le (1-t)np\right\} &\le e^{-t^2np/2}
\end{aligned}$$

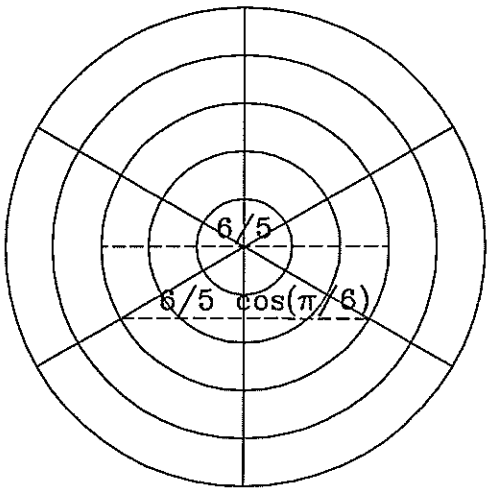

Figure 10: The 5-track 6-sector partitioning of the unit disk, $R_{5,3}$

In the proofs below, it is useful to subdivide the unit disk into smaller regions, and then reason about how many nodes will be placed into each of the regions. The track and sector subdivision of the unit disk is as follows. Let T, S be positive integers. A *T-track $2S$-sector* division of the unit radius disk is obtained by drawing T concentric circles, where circle i, $1 \leq i \leq T$, has radius i/T, and then drawing S straight lines through the center of these circles, where each successive line makes an angle of π/S radians with the previous line. Figure 10 shows a 5-track 6-sector division of the unit disk. Track i, $0 \leq i \leq T-1$, is the region between the circle of radius i/T and the circle of radius $(i+1)/T$.

Let $R_{T,S}$ be the set of regions created by the T-track $2S$-sector partitioning of the unit disk. Define $|r|$ to be the area of the region r, and let $|R|$ be the total area of all regions (π for the unit disk). For an instance I, let N_r be the set of all nodes which are located in region r.

For any region r, the probability that any one of the n nodes will be placed in the region is $\frac{|r|}{|R|}$, because the nodes are placed with a uniform distribution on the unit disk. The expected number of nodes in r is $n\frac{|r|}{|R|}$, and the expected total values of $\alpha(N_r)$ and $\omega(N_r)$ are both $n\frac{|r|}{|R|}\overline{\alpha}$.

We define an instance I to be ϵ, d*-balanced*, with respect to a set of regions R, if

$$\begin{array}{c} (\forall r \in R)\,[(1-\epsilon)\mathrm{E}(\alpha(N_r)) \;\leq\; \alpha(N_r) \;\leq\; (1+\epsilon)\mathrm{E}(\alpha(N_r))] \;\wedge \\ (\forall r \in R)\,[(1-\epsilon)\mathrm{E}(\omega(N_r)) \;\leq\; \omega(N_r) \;\leq\; (1+\epsilon)\mathrm{E}(\omega(N_r))] \;\wedge \\ (\exists v \in V)\left[\mathrm{dist}(v,C) \leq n^{-d}\right] \end{array} \tag{16}$$

where C is the center of the unit disk, and $\mathrm{dist}(v, C)$ denotes the Euclidean distance between the node v and the center.

LEMMA 3.3. *For instances I generated as described earlier, and all $0 < \epsilon < 1$, $0 < d < 1/2$, $T, S \geq 1$*

$$\lim_{n\to\infty} \Pr\{I \text{ is not } \epsilon, d\text{-balanced w.r.t. } R_{T,S}\} = 0$$

Proof:

$$
\begin{aligned}
&\Pr\{I \text{ is not } \epsilon, d\text{-balanced w.r.t. } R_{T,S}\} \\
&= \Pr\Big\{(\exists r \in R_{T,S})\,[\alpha(N_r) < (1-\epsilon)\mathrm{E}(\alpha(N_r)) \ \vee \ \alpha(N_r) > (1+\epsilon)\mathrm{E}(\alpha(N_r))] \ \vee \\
&\qquad (\exists r \in R_{T,S})\,[\omega(N_r) < (1-\epsilon)\mathrm{E}(\omega(N_r)) \ \vee \ \omega(N_r) > (1+\epsilon)\mathrm{E}(\omega(N_r))] \ \vee \\
&\qquad (\forall v \in V)\left[\mathrm{dist}(v,C) > n^{-d}\right]\Big\} \\
&\leq \sum_{r \in R_{T,S}} \Big[\Pr\left\{\alpha(N_r) < (1-\epsilon)n\frac{|r|}{|R|}\overline{\alpha}\right\} + \Pr\left\{\alpha(N_r) > (1+\epsilon)n\frac{|r|}{|R|}\overline{\alpha}\right\} \\
&\qquad\qquad + \Pr\left\{\omega(N_r) < (1-\epsilon)n\frac{|r|}{|R|}\overline{\alpha}\right\} + \Pr\left\{\omega(N_r) > (1+\epsilon)n\frac{|r|}{|R|}\overline{\alpha}\right\}\Big] \\
&\qquad + \left[1 - \frac{\pi n^{-2d}}{\pi}\right]^n
\end{aligned}
\tag{17}
$$

The equality holds by the definition of ϵ, d-balanced. The inequality holds because $\Pr\{A \vee B\} \leq \Pr\{A\} + \Pr\{B\}$, even if A and B are dependent events, as some pairs of events above happen to be.

Now we will find an upper bound for expressions of the form $\Pr\{A\}$, where A is either $\alpha(N_r) \leq (1-\epsilon)np\overline{\alpha}$ or $\alpha(N_r) \geq (1+\epsilon)np\overline{\alpha}$. We will temporarily substitute p for $\frac{|r|}{|R|}$ for readability.

$$
\begin{aligned}
\Pr\{A\} &= \sum_{k=0}^{n} \Pr\{|N_r| = k \wedge A\} && (18) \\
&\leq \sum_{k=0}^{\lfloor (1-\frac{\epsilon}{2})np \rfloor} \Pr\{|N_r| = k\} + \sum_{k=\lceil (1+\frac{\epsilon}{2})np \rceil}^{n} \Pr\{|N_r| = k\} \\
&\quad + \sum_{k=\lfloor (1-\frac{\epsilon}{2})np \rfloor + 1}^{\lceil (1+\frac{\epsilon}{2})np \rceil - 1} \Pr\{|N_r| = k \wedge A\} && (19) \\
&\leq \Pr\left\{|N_r| \leq (1-\frac{\epsilon}{2})np\right\} + \Pr\left\{|N_r| \geq (1+\frac{\epsilon}{2})np\right\} \\
&\quad + \epsilon np \max_k \Pr\{|N_r| = k \wedge A\} && (20)
\end{aligned}
$$

where the maximization in the last line is over the range $\lfloor (1-\frac{\epsilon}{2})np \rfloor + 1 \leq k \leq \lceil (1+\frac{\epsilon}{2})np \rceil - 1$. Step (18) holds because we are simply partitioning the event A into $n+1$ separate events. The inequality in step (19) holds because $\Pr\{|N_r| = k \wedge A\} \leq \Pr\{|N_r| = k\}$. The first line of (20) is just a rewritten form of the first line of (19), and the inequality follows from the maximization in the second line. The factor of ϵnp is an upper bound on the number of terms in the summation on the second line of (19).

By applying Theorem 3.4 with $t = \epsilon/2$, we see that the first line of (20) is at most

$$\begin{aligned} & \exp(-\epsilon^2 np/8) + \exp(-\epsilon^2 np/12) \\ \leq \; & 2\exp(-\epsilon^2 np/12) \end{aligned} \tag{21}$$

where the inequality follows simply because the second term is the larger of the two.

Now we will find an upper bound for the maximization term of (20), where A is $\alpha(N_r) \leq (1-\epsilon)np\overline{\alpha}$. If $|N_r| = k$, then let $N_r = \{v_1, \ldots v_k\}$

$$\begin{aligned} & \Pr\{|N_r| = k \wedge \alpha(N_r) \leq (1-\epsilon)np\overline{\alpha}\} \\ = \; & \Pr\left\{\sum_{i=1}^{k} \alpha(v_i) \leq k\overline{\alpha} - k\overline{\alpha}\left(1 - (1-\epsilon)\frac{np}{k}\right)\right\} && (22) \\ \leq \; & \exp\left(-2k\left(1-(1-\epsilon)\frac{np}{k}\right)^2 \overline{\alpha}^2/(\alpha_{\text{hi}} - \alpha_{\text{lo}})^2\right) && (23) \\ \leq \; & \exp\left(-2(1-\epsilon/2)np\left(1 - \frac{1-\epsilon}{1-\epsilon/2}\right)^2 \overline{\alpha}^2/(\alpha_{\text{hi}} - \alpha_{\text{lo}})^2\right) && (24) \\ = \; & \exp\left(-np\left(\frac{\epsilon^2}{2-\epsilon}\right)\overline{\alpha}^2/(\alpha_{\text{hi}} - \alpha_{\text{lo}})^2\right) && (25) \\ = \; & \exp\left(-npf(\epsilon)\right) && (26) \end{aligned}$$

Step (22) follows from rewriting and algebra. In step (23), we make use of Theorem 3.3 with $t = \overline{\alpha}(1-(1-\epsilon)np/k)$, which is larger than 0 because $k \geq (1-\epsilon/2)np$. Step (24) follows using the same lower bound on k, and step (25) follows from algebra. Step (26) is just a shorter form, where the function f has the obvious definition.

Using a similar derivation we may conclude

$$\begin{aligned} & \Pr\{|N_r| = k \wedge \alpha(N_r) \geq (1+\epsilon)np\overline{\alpha}\} \\ \leq \; & \exp\left(-np\frac{\epsilon^2(2-\epsilon)}{(2+\epsilon)^2}\overline{\alpha}^2/(\alpha_{\text{hi}} - \alpha_{\text{lo}})^2\right) \\ = \; & \exp\left(-npg(\epsilon)\right) && (27) \end{aligned}$$

We must use the upper bound on k, $k \leq (1+\epsilon/2)np$, for this derivation. Again, step (27) is for easier reading, and g is defined in the obvious way.

Now we may derive an upper bound on (17) by using (20), (21), (26), and (27).

$$\begin{aligned} & \sum_{r \in R_{T,S}} \left[8\exp\left(-\epsilon^2 n\frac{|r|}{12|R|}\right) + 2\epsilon n\frac{|r|}{|R|}\left(\exp\left(-nf(\epsilon)\frac{|r|}{|R|}\right) + \exp\left(-ng(\epsilon)\frac{|r|}{|R|}\right)\right)\right] \\ & \qquad + \left[1 - n^{-2d}\right]^n \\ \leq \; & 2ST\left[8\exp\left(-\epsilon^2 n/24ST^2\right) + 2\epsilon n\frac{2T-1}{2ST^2}\left(\exp\left(-nf(\epsilon)/2ST^2\right) + \exp\left(-ng(\epsilon)/2ST^2\right)\right)\right] \\ & \qquad + \exp\left(-n^{1-2d}\right) && (28) \\ = \; & c_1\exp(-c_2 n) + c_3 n\exp(-c_4 n) + c_3 n\exp(-c_5 n) + \exp\left(-n^{1-2d}\right) && (29) \end{aligned}$$

Step (28) follows because $\frac{1}{2ST^2} \leq \frac{|r|}{|R|} \leq \frac{2T-1}{2ST^2}$, which is true by the definition of the track and sector partition of the unit disk. The change in the last term is justified by the inequality $1+x \leq e^x$ for all real x. Step (29) is just a rewritten form of (28), where all c_i are positive real constants for any permissible values of ϵ, S, T, and any $\overline{\alpha} > 0$.

Since $0 < d < 1/2$, we can now easily see that this function's limit is 0 as n goes to infinity. ∎

LEMMA 3.4. *Let I be an instance which is ϵ, d-balanced with respect to the partition $R_{T,S}$. Then the lower bound $L(I)$ for this instance satisfies*

$$L(I) \geq 2(1-\epsilon)n\overline{\alpha}\cos\left(\frac{\pi}{2S}\right)\frac{2(T+1/4)(T-1)}{3T^2} \tag{30}$$

Proof: The lower bound is equal to the cost of a maximum cost flow in the lower bound network. This proof will be done by constructing a flow, not necessarily of maximum cost, which costs at least as much as the right hand side of Inequality (30). From this it follows that $L(I)$ also satisfies the inequality.

In the track and sector partitioning $R_{T,S}$, let r and r' be two regions which are in the same track i, but in "opposite" sectors (i.e., going around track i from r to r' in either direction, we encounter $S-1$ other sectors before reaching r').

In the lower bound network $\mathcal{L}$ for this instance, there are vertices u_s for each $u \in N_r$ and vertices u'_d for each $u' \in N_{r'}$. Because the traffic is α, ω-bounded, we can send $\min\{\alpha(N_r), \omega(N_{r'})\}$ flow from all of the source vertices to all of the destination vertices. Since I is ϵ, d-balanced, this quantity is at least

$$\begin{aligned}(1-\epsilon)\mathrm{E}(\alpha(N_r)) &= (1-\epsilon)n\frac{|r|}{|R|}\overline{\alpha} &(31)\\ &= (1-\epsilon)n\frac{2i+1}{2ST^2}\overline{\alpha} &(32)\end{aligned}$$

Each of the arcs in the lower bound network has a cost equal to the distance between the nodes. The distance between any node in r and any node in r' is at least $(2i/T)\cos(\pi/2S)$. See Figure 10 for a visual example of why the distance can be smaller than $2i/T$.

Therefore, we can make a flow from the vertices u_s to the vertices u'_d which costs at least

$$\left((1-\epsilon)n\frac{2i+1}{2ST^2}\overline{\alpha}\right)\left(\frac{2i}{T}\cos\left(\frac{\pi}{2S}\right)\right) \tag{33}$$

A similar flow can be set up between nodes in all pairs of opposite sectors. The total cost of such a flow is obtained by summing Equation (33) over all sectors.

$$\begin{aligned}&\sum_{i=0}^{T-1} 2S(1-\epsilon)n\frac{2i+1}{2ST^2}\overline{\alpha}\frac{2i}{T}\cos\left(\frac{\pi}{2S}\right)\\ &= 2(1-\epsilon)n\overline{\alpha}\cos\left(\frac{\pi}{2S}\right)\frac{1}{T^3}\sum_{i=0}^{T-1} i(2i+1)\\ &= 2(1-\epsilon)n\overline{\alpha}\cos\left(\frac{\pi}{2S}\right)\frac{2(T+1/4)(T-1)}{3T^2}\end{aligned}$$

where the equalities follow from algebraic manipulation and the identities $\sum_{i=0}^{n} i = n(n-1)/2$ and $\sum_{i=0}^{n} i^2 = n(n-1/2)(n-1)/3$. ∎

LEMMA 3.5. *Let I be an instance which is ϵ, d-balanced with respect to the partition $R_{T,S}$. Then the cost $A(I)$ of the minimum cost star network satisfies*

$$A(I) \leq 2(1+\epsilon)n\overline{\alpha}\left[\frac{2(T-1/4)(T+1)}{3T^2} + n^{-d}\right] \tag{34}$$

Proof: The proof will be done by constructing a star network, not necessarily of minimum cost, that costs no more than the expression on the right hand side of Inequality (34). From this it follows that $A(I)$ satisfies the inequality.

Since I is ϵ, d-balanced, there is a node C which is no further than n^{-d} from the center of the disk.

Consider a sector r in track i of $R_{T,S}$. For each node $u \in N_r$, we can build a link (u, C) with capacity $\alpha(u)$ and a link (C, u) with capacity $\omega(u)$, and the tree network will be nonblocking. The distance from C to u is at most $(i+1)/T + n^{-d}$. Therefore the contribution to the total network cost from nodes in r is at most

$$\begin{aligned}
&(\alpha(N_r) + \omega(N_r))\left(\frac{i+1}{T} + n^{-d}\right) \\
&\quad \leq \quad 2(1+\epsilon)\mathrm{E}(\alpha(N_r))\left(\frac{i+1}{T} + n^{-d}\right) \\
&\quad = \quad 2(1+\epsilon)n\frac{2i+1}{2ST^2}\overline{\alpha}\left(\frac{i+1}{T} + n^{-d}\right)
\end{aligned} \tag{35}$$

because I is ϵ, d-balanced. Summing Equation (35) over all sectors we get

$$\begin{aligned}
&\sum_{i=0}^{T-1} 2S \cdot 2(1+\epsilon)n\frac{2i+1}{2ST^2}\overline{\alpha}\left(\frac{i+1}{T} + n^{-d}\right) \\
&\quad = \quad 2(1+\epsilon)n\overline{\alpha}\frac{1}{T^3}\sum_{i=0}^{T-1}(2i+1)(i+1+Tn^{-d}) \\
&\quad = \quad 2(1+\epsilon)n\overline{\alpha}\left[\frac{2(T-1/4)(T+1)}{3T^2} + n^{-d}\right]
\end{aligned}$$

where the equalities follow from algebraic manipulation and the same identities as used in the proof of Lemma 3.4. ∎

LEMMA 3.6. *Let I be an instance which is ϵ, d-balanced with respect to the partition $R_{T,S}$. Then for any $\delta > 0$,*

$$\frac{A(I)}{L(I)} \leq 1 + \delta \tag{36}$$

if $\epsilon \leq \frac{x-1}{x+1}$, $S \geq \frac{\pi}{2\cos^{-1}(1/x)}$, $T \geq \max\{2, \frac{3(x+3)}{4(x-1)} + \frac{x-1}{2}\}$, and $n \geq \left[\frac{6}{x-1}\right]^{1/d}$, where $x = (1+\delta)^{1/3}$.

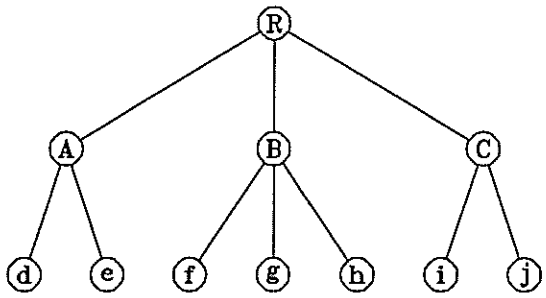

Figure 11: An example hierarchy tree

Proof: Since I is ϵ, d-balanced, then by Lemmas 3.4 and 3.5 and algebra we have

$$\frac{A(I)}{L(I)} \leq \left(\frac{1+\epsilon}{1-\epsilon}\right) \frac{1}{\cos(\pi/2S)} \left[\frac{(T-1/4)(T+1)}{(T+1/4)(T-1)} + \frac{3T^2}{2(T+1/4)(T-1)} n^{-d}\right] \quad (37)$$

It is easy to verify that $\epsilon \leq \frac{x-1}{x+1}$ implies $\frac{1+\epsilon}{1-\epsilon} \leq x$. Similarly, $S \geq \frac{\pi}{2\cos^{-1}(1/x)}$ implies $\frac{1}{\cos(\pi/2S)} \leq x$.

$T \geq \frac{3(x+3)}{4(x-1)} + \frac{x-1}{2}$ implies $\frac{(T-1/4)(T+1)}{(T+1/4)(T-1)} \leq (x+1)/2$, and $T \geq 2$ and $n \geq \left[\frac{6}{x-1}\right]^{1/d}$ imply $\frac{3T^2}{2(T+1/4)(T-1)} n^{-d} \leq (x-1)/2$.

All together, these conditions imply that the right hand side of Equation (37) is at most

$$\begin{aligned} x \cdot x \cdot [(x+1)/2 + (x-1)/2] &= x^3 \\ &= 1+\delta \end{aligned}$$

■

4. Hierarchical traffic limits

4.1. Definition

The nonblocking traffic limits defined in section 3 give a good way of specifying the traffic for a "flat" network, with no additional structure. This method can be easily extended to specify traffic in a hierarchical way, where there are "clusters" of nodes which may have high traffic among themselves, but less traffic between nodes in the cluster and nodes outside of the cluster. These clusters may come about because the network owners charge the users more to establish connections outside of their cluster, because there are groups of users who communicate more frequently among themselves than with others on the network, or some combination of these causes.

As an example, Figure 11 shows the hierarchical structure $\mathcal{H}$ of an instance with 7 nodes, d through j, which are organized into 3 clusters, A, B, and C. R is the "root" cluster, which

u	$\alpha(u)$	$\omega(u)$
d	5	5
e	6	6
f	3	9
g	4	4
h	7	7
i	8	8
j	5	2
A	6	6
B	7	8
C	7	5

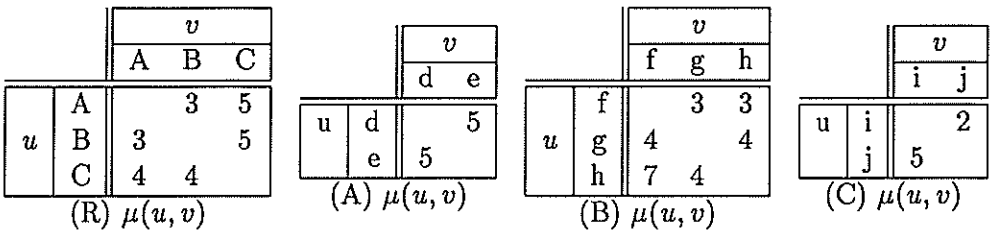

Figure 12: Example hierarchical traffic limits

is only used as a place holder so that the hierarchical structure may be drawn as a rooted tree. We say that a node u is in the cluster v if u is a descendant of v in the hierarchy tree $\mathcal{H}$. We will denote the set of clusters by $\mathcal{C}$ ($\mathcal{C}$ does not contain the root cluster R, however).

For the moment, ignore point-to-point restrictions on the traffic, and let us only consider termination capacities. Figure 12 gives α and ω values for all of the terminal nodes d through j. If the traffic were flat, then there could simultaneously exist connections with total rate up to $\alpha(f) + \alpha(g) + \alpha(h) = 3 + 4 + 7 = 14$ which have their source as one of f, g, h and their destination as one of d, e, i, j. However, suppose we know that there will never be such connections with total rate more than 7 at any given moment. This can be specified by saying that the source termination capacity of the cluster B, $\alpha(B)$, is 7. Similarly, let the maximum total rate of connections with source outside of B and destination in B be 8, and call this the destination termination capacity of cluster B, $\omega(B)$. Similar values for clusters A and C are given in Figure 12.

Given this form of aggregating nodes into clusters, it now makes less sense to attempt specifying a point-to-point restriction for every pair of terminal nodes. Furthermore, the lower bound network which we will define shortly cannot specify such detailed restrictions.

It does make sense to specify point-to-point restrictions between nodes in the same cluster, and also between clusters. The lower bound network can easily handle these restrictions. For example, since $\alpha(B) = 7$ and $\omega(A) = 6$, we know that the total rate of connections with source in B and destination in A can never be more than $\min\{7, 6\} = 6$. If we also knew that nodes in clusters B and A communicate very infrequently, we could specify a point-to-point restriction from cluster B to A, denoted $\mu(B, A)$, of less than 6, say $\mu(B, A) = 3$. Also specify $\mu(A, B) = 3$. Figure 12(A) specifies μ values for all pairs of nodes which are both children of A in the hierarchy tree (Figure 11). Similarly, Figures 12(B), (C), and (R) do the same for all pairs of children of vertices B, C, and R.

The example constructed in this section is a two-level hierarchy. One can think of flat traffic as a one-level hierarchy, where the hierarchy tree contains a vertex for each network node, plus the root R, and all nodes are children of R.

This method of specifying traffic can be used for any hierarchical structure which can

be given by a hierarchy tree like the one in Figure 11. In general, all leaves of the tree are terminal nodes with α and ω values, all other tree vertices except R are clusters with their own α and ω values, and μ values may be specified between every pair of tree vertices which have a common parent vertex. We will denote hierarchical traffic limits $\mathcal{T}$ by the tuple $(\mathcal{H}, \alpha, \omega, \mu)$, where the functions α, ω, and μ have the appropriate domains.

Given a network $\mathcal{N} = (N, L, cap)$ and a (multi)set of connection requests $\mathcal{R}$, we can extend the definitions of the *point-to-point usage*, *source usage*, and *destination usage*, given in section 3.1, to include clusters $u, v \in \mathcal{C}$ as well as terminal nodes. In the following definitions, we consider a cluster $u \in \mathcal{C}$ to be the set of all nodes in N which are contained in the cluster u.

$$\begin{aligned} \textit{src-usage}(u, \mathcal{R}) &= \sum_{q \in \mathrm{R},\, u \cap src(q) \neq \emptyset} rate(q) \\ \textit{dest-usage}(u, \mathcal{R}) &= \sum_{q \in \mathrm{R},\, u \cap dest(q) \neq \emptyset} rate(q) \\ \textit{pp-usage}(u, v, \mathcal{R}) &= \sum_{q \in \mathrm{R},\, u \cap src(q) \neq \emptyset,\, v \cap dest(q) \neq \emptyset} rate(q) \end{aligned}$$

Let hierarchical traffic limits $\mathcal{T} = (\mathcal{H}, \alpha, \omega, \mu)$ be given. We say that the set of requests $\mathcal{R}$ *is compatible with traffic limits* $\mathcal{T}$ if the following condition holds.

$$\begin{gathered} (\forall u \in N \cup \mathcal{C})\ (\textit{src-usage}(u, \mathcal{R}) \leq \alpha(u)) \wedge \\ (\forall u \in N \cup \mathcal{C})\ (\textit{dest-usage}(u, \mathcal{R}) \leq \omega(u)) \wedge \\ (\forall u, v \text{ which are siblings in } \mathcal{H})\ (\textit{pp-usage}(u, v, \mathcal{R}) \leq \mu(u, v)) \end{gathered} \tag{38}$$

That is, no node or cluster is involved in more requests than its termination capacity will allow, and no pair of nodes/clusters is involved in more requests than their point-to-point restriction will allow.

4.2. Extending the lower bound

Now we will generalize the definition of the lower bound network $\mathcal{L} = (V, E, cap)$ for hierarchical traffic limits. The edges are given in the form (u, v, c) where the edge is from vertex u to v and has capacity c.

$$\begin{aligned} V &= \{s, t\} \cup \{u_{s1}, u_{d1} : u \in N\} \cup \{u_{s1}, u_{s2}, u_{d1}, u_{d2} : u \in \mathcal{C}\} \\ E &= \{(s, u_{s1}, \alpha(u)),\ (u_{d1}, t, \omega(u)) : u \in N\} \cup \\ &\quad \{(u_{s2}, u_{s1}, \alpha(u)),\ (u_{d1}, u_{d2}, \omega(u)) : u \in \mathcal{C}\} \cup \\ &\quad \{(u_{s1}, v_{s2}, \infty),\ (v_{d2}, u_{d1}, \infty) : u \text{ a child of } v,\ v \neq R\} \cup \\ &\quad \{(u_{s1}, v_{d1}, \mu(u, v)) : u, v \text{ are siblings in } \mathcal{H}\} \end{aligned}$$

The lower bound network for the hierarchical traffic limits of Figure 12 is shown in Figure 13. The first number labeling each edge is the edge's capacity. The unlabeled edges have infinite capacity. The other numbers will be explained later. All edges are directed from left to right.

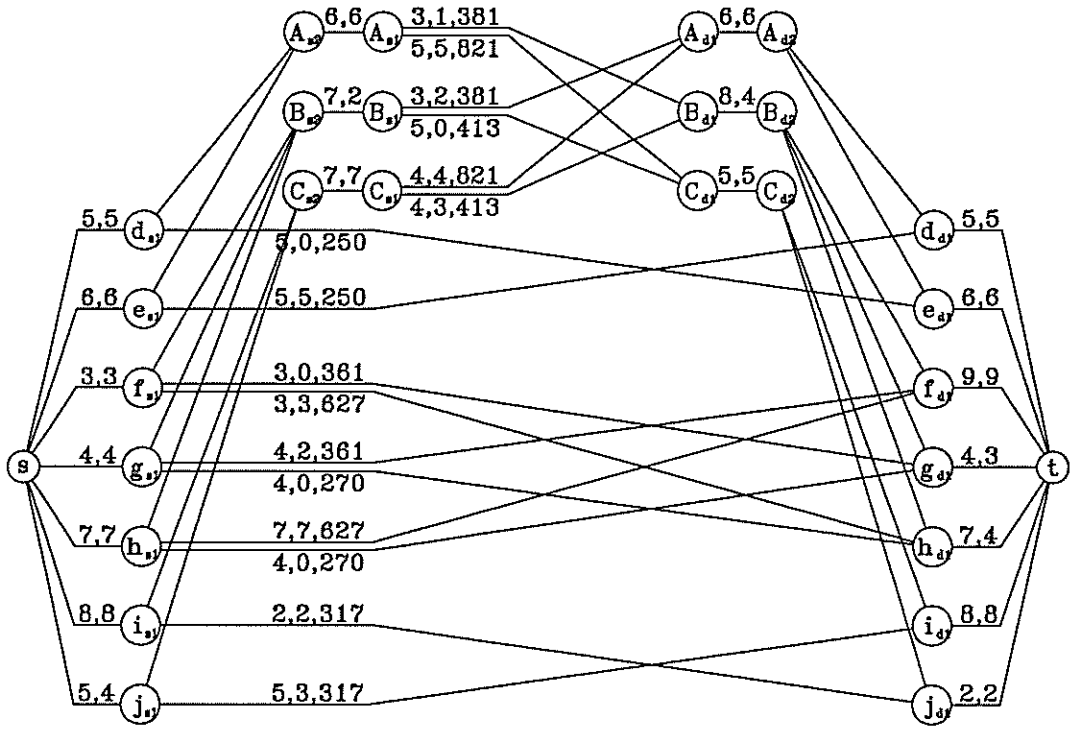

Figure 13: Lower bound network for traffic limits of Figure 12

Just as for flat traffic limits, there is a many to one correspondence from compatible sets of point-to-point connection requests to integer flows f in $\mathcal{L}$ which do not exceed any edge capacities.

The method of section 3.2 can be extended to find lower bounds on the costs of non-blocking networks with hierarchical traffic limits. It can be done by finding a maximum cost flow in the lower bound network as before, but because of restrictions on the costs that we may assign to edges, the lower bounds are not as high as possible; there are portions of the lower bound cost that cannot be counted using maximum cost flows on the lower bound network. This difficulty can be avoided by using a linear programming formulation, giving better lower bounds at the cost of more computation time.

First we extend the maximum cost flow method. Consider the example hierarchical lower bound network of Figure 13. It should be clear from the previous section that we should define the costs of all edges of the form (s, u_{s1}) and (v_{d1}, t) to be 0, and the costs of all edges of the form (u_{s1}, v_{d1}) to be $\gamma(u, v)$, where u, v are both nodes in N (e.g., edges (d_{s1}, e_{d1}) and (j_{s1}, i_{d1}), but not (A_{s1}, B_{d1})). However, how should we define the other edge costs?

One property that should hold for our assignment of edge costs is: for any terminal node pair u, v, the total cost of all edges on any path from u_{s1} to v_{d1} should be at most $\gamma(u, v)$. We will call this the *valid cost* property, and any path which violates it is called *invalid.* If the valid cost property is violated, we are not guaranteed that the cost of a flow will be a

u	d	e	f	g	h	i	j	Position
d	0	250	539	448	503	922	855	(100,700)
e		0	702	472	381	971	821	(250,900)
f			0	361	627	510	633	(300,200)
g				0	270	500	413	(500,500)
h					0	681	461	(600,750)
i						0	317	(800,100)
j							0	(900,400)

Figure 14: Link costs for hierarchical example

lower bound for the cost of any nonblocking network. Any flow on an invalid path may cost more than is actually required to build the necessary links.

Given the restriction of keeping the costs valid, we want to define the costs so that the maximum cost flow will have a cost as large as possible. It appears that computing these optimum edge costs exactly would require solving a nonlinear optimization problem. Even if this task could be done efficiently, there are instances of hierarchical network design problems for which the costs between nodes in different clusters cannot be accurately modeled in the lower bound network. However, there is a simple way to define edge costs which usually seems to do fairly well (from the experimental results presented later), and they can be determined very easily.

The basic idea is to examine some of the edges of the lower bound network in a given order. On each step, increase the cost of the current edge as much as possible while still maintaining the validity of the edge costs. An approach used in the experimental results is to consider the edges (u_{s1}, v_{d1}), where u, v range over all sibling pairs of nodes in the hierarchy tree. If only these edges are considered, then the order of their consideration does not matter. The resulting edge costs will be the same in any case.

For the example hierarchical instance given in Figures 11 and 12, let the link cost coefficients $cost(\cdot)$ be given by the entries of the table in Figure 14. The entries are symmetrical, and equal to the euclidean distance between the nodes (rounded up), where the position of each node is given in the table. Valid edge costs are given by the third number labeling each edge in Figure 13. If there is no edge label, or only two numbers labeling the edge, then the cost is 0. One can see that the cost of edge (A_{s1}, B_{d1}) is 381, which is the minimum of the costs from any node in cluster A to any node in cluster B. This is the largest valid cost the edge may have. Therefore traffic between some pairs of nodes in those clusters is not charged as much as it could be.

Once the maximum cost flow in $\mathcal{L}$ has been computed, the cost of this flow is a lower bound. In Figure 13, a maximum cost flow is given by the second number labeling each edge. Its cost is 19598, which is a lower bound on the cost of any nonblocking network.

Even better, we may find any set of compatible point-to-point requests $\mathcal{R}$ which corresponds to the maximum cost flow f, and a lower bound is $\sum_{(u,v,r)\in\mathcal{R}} \gamma(u,v)\cdot r$. This value is at least as large as the cost of f, because the edge costs must be valid. In most cases it will

be larger than the cost of f, and it will be used to compute the lower bounds for the experimental results in section 4.5. For example, one set of requests which corresponds to the maximum cost flow of Figure 13 is $\mathcal{R} = \{(d,g,1),(d,i,4),(e,i,1),(e,d,5),(f,h,3),(g,e,2),(g,f,2),(h,f,7),(i,e,4),(i,g,2),(i,j,2),(j,h,1),(j,i,3)\}$. The cost implied by this set of requests is 21223, which is significantly higher than the cost of the flow itself.

The best lower bound known to the author can be obtained by using a linear programming formulation. A maximum flow problem can represent some problems that linear programs can, but linear programs can represent many more problems. In this linear program, we will use variables $x_{u,v}$, where $u, v \in N, u \neq v$. Variable $x_{u,v}$ represents the total traffic from u to v. A collection of linear inequalities will represent the restrictions on the $x_{u,v}$ which are imposed by the traffic limits. In this linear program, there will be a many to one correspondence from compatible sets of point-to-point requests $\mathcal{R}$ to feasible integer solutions x, similar to the correspondence for flows in $\mathcal{L}$.

To ease the presentation, we will use the notation $x_{Y,Z}$, where Y, Z are subsets of N, to represent $\sum_{u \in Y, v \in Z} x_{u,v}$. A single node u will represent the set $\{u\}$, and a cluster c will represent the set of nodes within the cluster. The complement of a set Y, $N - Y$, will be denoted by $\overline{Y}$.

Given these definitions, we may define the inequalities very similarly to the definition of compatible request sets, condition (38). They are:

$$\begin{array}{lcll} x_{u,\overline{u}} & \leq & \alpha(u) & \forall u \in N \cup C \\ x_{\overline{u},u} & \leq & \omega(u) & \forall u \in N \cup C \\ x_{u,v} & \leq & \mu(u,v) & \forall u, v \text{ which are siblings in } \mathcal{H} \\ x_{u,v} & \geq & 0 & \forall u, v \in N, u \neq v \end{array} \tag{39}$$

The linear program is:

$$\begin{array}{rl} \text{Maximize} & \sum_{u \neq v} \gamma(u,v) \cdot x_{u,v} \\ \text{Subject to:} & \text{Inequalities (39)} \end{array}$$

For our hierarchical example, an optimal solution to the linear program is $x_{de} = 4, x_{dj} = 1, x_{ed} = 1, x_{ef} = 1, x_{ei} = 4, x_{fh} = 3, x_{gd} = 2, x_{gf} = 1, x_{gh} = 1, x_{hf} = 7, x_{id} = 2, x_{ie} = 2, x_{ih} = 3, x_{ij} = 1, x_{ji} = 4$, giving a lower bound of 21902. This is 3.2% higher than the lower bound found using the maximum cost flow method.

4.3. Link dimensioning in tree-shaped networks

Link dimensioning in a tree-shaped network can be done using the same method as in section 3.3.2, based on finding maximum flows in modified lower bound networks.

4.4. Computing inexpensive hierarchical star networks

For hierarchical traffic, we generalize the notion of a star network to a hierarchical star network. We go through the hierarchy tree of the instance in a bottom-up fashion, computing the cheapest star subnetwork for each of the lowest level clusters. Then treat each of the lowest level clusters as a single node located at the center of the chosen star subnetwork, find the cheapest star subnetwork for the next to lowest level clusters.

As before, we can precompute the minimum necessary link capacities in and out of each node, instead of performing a redundant calculation for each star subnetwork tried.

For example, consider the hierarchical instance described in Figures 11, 12, and 14. For clusters A and C, the cheapest star subnetwork is the one with two links, one in each direction, between the nodes in the cluster. Arbitrarily choose node e as the center of A's subnetwork, and i as the center of C. For cluster B, the cheapest subnetwork is obtained by choosing g as the center. Now we find the cheapest star network connecting the subnetworks, where we treat all nodes in A as if they are located at d, all nodes in B at g, and all nodes in C at i. This network is the one with center g, so we add links $\{(d,g),(g,d),(i,g),(g,i)\}$ to the star subnetworks already found. The cost of this network is 24495, which is 11.8% over the linear programming lower bound of 21902, and 15.4% over the maximum cost flow lower bound of 21223. Therefore this network costs at most 11.8% over optimal.

This procedure can be implemented to run using $2(|N|+|\mathcal{C}|)$ maximum flow computations, each requiring only $O(n)$ time due to their special structure. The best centers for each cluster can be determined in time that is linear in the number of edges of the hierarchy tree.

4.5. Experimental results for hierarchical traffic

In this section we present experimental results comparing inexpensive hierarchical star network costs, computed as described in section 4.4, to the lower bounds. Here the lower bounds are computed in two ways. First there is the faster but less accurate method of using the lower bound network with valid edge costs, described in Section 4.2, and then the slower but more accurate method of solving a linear program, described later in the same section.

All instances generated are two-level hierarchies, although the methods for instance generation and all algorithms work for arbitrary hierarchical structures.

To generate random two-level hierarchies, we are given a number of clusters c and a number of nodes m to place within each cluster, giving a total of $n = cm$ nodes. In addition to the the parameters given for flat instances in section 3.4, we are given a range of real numbers $[\alpha^c_{lo}, \alpha^c_{hi}]$, which is a sub-interval of $[0,1]$.

When placing nodes, we assume that not only do clusters have higher traffic among the nodes within, but the nodes are also geographically clustered together. For each cluster, we generate a random width w in the interval $[.1,.25]$ and a height h in the same interval. A bounding rectangle for the cluster with these dimensions is then randomly placed in the

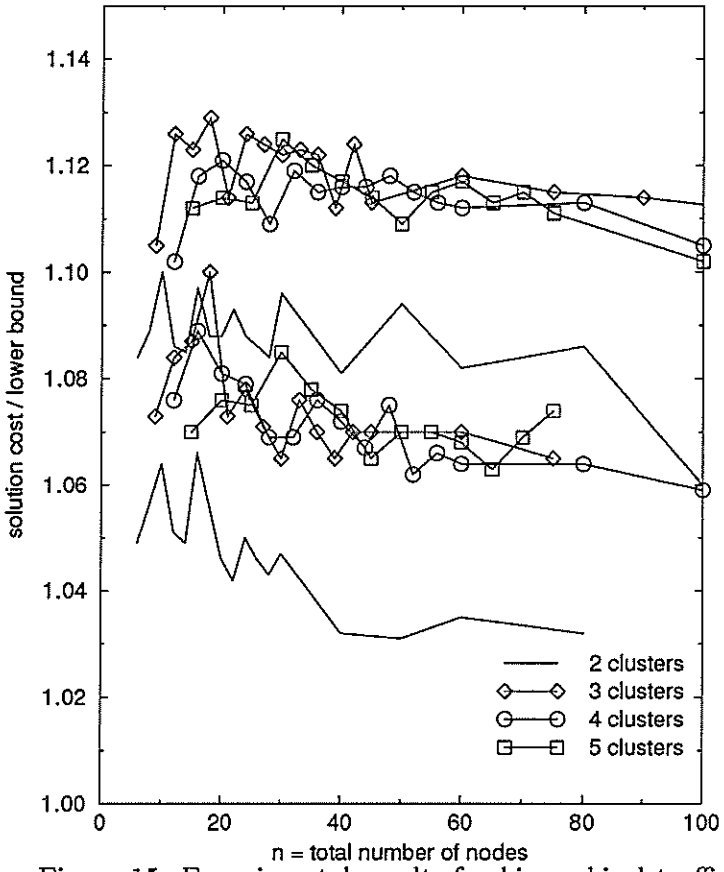

Figure 15: Experimental results for hierarchical traffic

unit square, and all m nodes within the cluster are randomly placed within this bounding rectangle. Bounding rectangles for different clusters are allowed to overlap. Link cost coefficients $\gamma(u, v)$ are set equal to the Euclidean distance between u and v. α values for nodes (but not clusters) are generated as before. For each cluster U containing nodes $u_1, \ldots u_m$, generate a random real value x in the interval $[\alpha^c_{lo}, \alpha^c_{hi}]$ and set $\alpha(U) = \omega(U) = x \cdot \sum_{i=1}^{m} \alpha(u_i)$. If this interval is $[1, 1]$ then the clusters do not constrain source and termination capacity any more than that of the constituent nodes (although μ values for the clusters may constrain the traffic more than if there were no clusters). All experiments reported here use the interval $[.1, .3]$, so that 10% to 30% of a cluster's traffic may be between nodes within it and nodes in other clusters.

The top four curves of Figure 15 show the average ratio of the hierarchical star network cost over the network lower bound. Each curve represents a different value of c, the number of clusters, as shown. The number of nodes per cluster m ranges over the same values as n did in the previous section, as long as $n = cm \leq 100$. Each data point is the average performance ratio of 80 randomly generated instances. Experiments were also run for up to $c = 10$ clusters, but the curves for more than 5 clusters do not differ significantly from those with 3 to 5 clusters, so they are not shown.

It seems that there may be a trend towards 1, but if so, then it occurs much more slowly than before. The performance ratios are significantly better when there are two clusters, as

opposed to three or more. If we had flat instances with two nodes, the lower bound and the minimum cost solution have exactly the same value. It is only for three or more nodes that they can differ. Similarly, when we have two level hierarchical instances with two clusters, the portion of the lower bound caused by the inter-cluster traffic is close to the cost of the links between the clusters. When there are three or more clusters, these values can differ significantly. We expect that curves for large numbers of clusters (say 50) would be closer to the $c = 2$ curves than to the curves for 3,4, or 5 clusters.

The lower four curves of Figure 15 show the average ratio of the hierarchical star network cost over the linear programming lower bound. Each data point is the average ratio of 30 randomly generated instances, which are identical to 30 of the instances used in generating the upper four curves. The linear programming lower bound was not computed for the larger instances, because of the time and memory required to compute them. The largest instances for which the linear program was solved required up to 3 hours of CPU time on a Sun 4/110 workstation, and approximately 15 megabytes of virtual memory. They were solved using an unsophisticated (but free) implementation of the simplex method, and both the time and memory required could probably be reduced by a factor of 5 to 10 for more careful implementations taking advantage of the sparsity of non-zero coefficients. Again, experiments were run for up to 10 clusters, but the curves for more than 5 clusters are very similar to those for 3 to 5 clusters.

The range of values is notably lower now, from about 1.03 up to 1.10. There seems to be more of a tendency to approach 1 as n increases. Thus we should also be able to extend the probabilistic result of Section 3.6 to some kinds of hierarchical traffic limits as well.

5. Generalizations

5.1. Link dimensioning in general networks

When the union of the path edges in the table *path* is not a tree-shaped network, then a more general method than the one in sections 3.3 and 4.3 is required to compute $maxusage(l)$.

If it is true, as we conjecture, that tree-shaped networks are cheapest among all non-blocking networks, then the algorithms of this section may not be beneficial to implement for actual use in network configuration. However, these results will be necessary to prove that conjecture.

In general, for any link l and table *path*, define a set of node pairs $P_l = \{(u, v) : l \in path(u, v)\}$. This is the set of node pairs which must use link l when they communicate. In section 3.3.2 we saw that for tree-shaped networks the set P_l was always of the form $\{(u, v) : u \in X_l, v \in N - X_l\}$. In fact, the method of that section works whenever P_l is of the form $\{(u, v) : u \in Y, v \in Z\}$, where $Y, Z \subseteq N$ are any sets of nodes. However, when P_l does not have this special structure, those methods do not work.

First we will handle the case of flat traffic limits. Consider the lower bound network $\mathcal{L}$. If u, v are any distinct pair of switches such that $(u, v) \notin P_l$, then there will be no connections from u to v which use link l. We may model this by reducing the point-to-point restriction

$\mu(u,v)$ to 0. That is, reduce the capacity of the edge (u_s, v_d) in $\mathcal{L}$ to 0. Equivalently, that edge may be removed. Again, call this modified lower bound network $\mathcal{L}_l$.

Now there is a many to one correspondence from sets $\mathcal{R}_l$ to integer flows in $\mathcal{L}_l$, where $\mathcal{R}$ is a compatible set of point-to-point requests. Thus we have reduced the problem of determining *maxusage*(l) to a maximum flow problem, as in section 3.3.2.

For hierarchical traffic limits, however, there may be no way to modify the edge capacities in $\mathcal{L}$ such that this many to one correspondence holds. For example, recall the example hierarchical traffic limits of section 4.1 and the lower bound network of Figure 12. Suppose that for some link l we have $P_l = \{(d,i),(d,j),(e,i)\}$. The difficulty is preventing traffic from going between the pair (e,j) while allowing traffic between the pairs in P_l. Both d and e can be a source for traffic, and both i and j can receive it, so we cannot accurately model this situation by reducing the α or ω of the nodes involved. Reducing any of the capacities of edges on the path $A_{s2} - A_{s1} - C_{d1} - C_{d2}$ prevents the pair (e,j) from communicating, as we desire, but it also prevents the pairs in P_l from doing so.

This difficulty can be avoided by modifying the linear programming formulation given in section 4.2 to compute *maxusage*(l). For any pair (u,v) which is not in P_l, add the restriction $x_{u,v} = 0$. Equivalently, we may remove $x_{u,v}$ from the formulation completely, which would make the linear program smaller and more efficient to solve using general linear programming algorithms. The objective function to maximize is the total traffic, which is the sum of all variables $x_{u,v}$.

The complete linear program LP_l is then

$$\begin{array}{lll} LP_l: & \text{Maximize} & \sum_{u \neq v} x_{u,v} \\ & \text{Subject to:} & \text{Inequalities (39)} \\ & & x_{u,v} = 0 \qquad\qquad (u,v) \notin P_l \end{array}$$

As for flows in section 3.3.2, there is a many to one correspondence from sets $\mathcal{R}_l$ to integer solutions x of LP_l, where $\mathcal{R}$ is a compatible set of point-to-point requests. Therefore

$$\begin{aligned} \textit{maxusage}(l) &= \max_{\mathcal{R} \in ALLREQ_T} \left(\sum_{q \in \mathcal{R}_l} \textit{rate}(q) \right) \\ &= \max_{x \in INT_l} \left(\sum_{u \neq v} x_{u,v} \right) \qquad (40) \\ &= \max_{x \in REAL_l} \left(\sum_{u \neq v} x_{u,v} \right) \end{aligned}$$

where INT_l is the set of all integer solutions to LP_l, and $REAL_l$ is the set of all real solutions. The first line is just Equation (4), and (40) follows from the many to one correspondence from request sets $\mathcal{R}$ to integer solutions x. The last line holds because all basic feasible solutions of LP_l are integral. This is because the coefficient matrix which arises from the inequalities is totally unimodular, and all basic feasible solutions of such a linear program are integral when the right-hand side values (here the α, ω, μ values) are integral [PS82, Section 13.2].

5.2. Handling multipoint traffic

In every previous section on link dimensioning, we have only explicitly considered point-to-point requests and connections. If allowing multipoint connections could increase $maxusage(l)$ in some situations, then we would want to find a way to compute these larger values. In this section, we show that $maxusage(l)$ remains the same when multipoint connections are allowed. This means that the worst-case traffic pattern for any link can be achieved by point-to-point connections only.

Fingerhut [Fin92, Section 7.1.2] proves that the link capacities $cap(l) = maxusage(l)$, for all $l \in L$, are nonblocking when multicast connection requests $(u, \{v_1, \ldots, v_k\}, r)$ are satisfied by using a route which is a subtree of $\bigcup_{i=1}^{k} path(u, v_i)$. This was proved for flat traffic limits in arbitrary networks. We will restate the proof and extend it to hierarchical traffic as well.

There are very similar arguments for the case when we have a lower bound network $\mathcal{L}_l$, and when we must use the linear program LP_l. We give the argument for $\mathcal{L}_l$ below, and occasionally add parenthesized phrases, which are the minor changes necessary for the proof to work for LP_l.

The basic idea of the argument is to show that the many to one correspondence from sets $\mathcal{R}_l$, to integer flows f in $\mathcal{L}_l$ (integer solutions x of LP_l), where $\mathcal{R}$ is a compatible set of point-to-point requests, can be extended to allow multipoint requests in $\mathcal{R}$ as well. This correspondence should preserve the property that $\sum_{q \in \mathcal{R}_l} rate(q)$ is equal to the value of the the flow f (the value of the objective function for x). If we can do this, then steps (5) and (40) of the previous derivations will still hold when $ALLREQ_{\mathcal{T}}$ is replaced with $ALLREQ'_{\mathcal{T}} = \{\mathcal{R} : \mathcal{R} \text{ is compatible with } \mathcal{T}\}$. This the set of all request sets which are compatible with the traffic $\mathcal{T}$, not just the point-to-point ones.

Let the nodes N be ordered by assigning them unique numbers from 1 to n. The particular order chosen is unimportant; it is a way to make the correspondence many to one as desired.

Given a compatible request set $\mathcal{R}$, we must show how to construct a unique integer flow f in $\mathcal{L}_l$ (solution x of LP_l). Start with $f = 0$ ($x = 0$). For any requests $q \in \mathcal{R}$ which are not in $\mathcal{R}_l$, the connection which realizes q will not use link l, so do nothing. For point-to-point requests $q = (u, v, r)$ which are in $\mathcal{R}_l$, link l must be utilized, so "charge" for it by adding flow r on the path $s - u_s - \ldots - v_d - t$, where the dots represent the unique path from u_s to v_d (add r to x_{u_s,v_d}). For one to many multipoint requests $q = (u, \{v_1, \ldots, v_k\}, r)$, it is possible that link l will be used in the connection, so add flow r on the path $s - u_s - \ldots - w_d - t$, where w is the smallest numbered node in $\{v_1, \ldots, v_k\}$ such that $(u, w) \in P_l$ (add r to x_{u_s,w_d}). This means we are assuming that l is being used in the connection if l is in $\bigcup_{i=1}^{k} path(u, v_i)$, even though it may not be used because only a subtree of that link set is used. This does not invalidate the proof; we are simply being pessimistic about the usage of links with multipoint connections. Similarly, for many to one multipoint requests $q = (\{v_1, \ldots, v_k\}, u, r)$, add flow r on the path $s - w_s - \ldots - u_d - t$, where w is the smallest numbered node in $\{v_1, \ldots, v_k\}$ such that $(u, w) \in P_l$ (add r to x_{w_s,u_d}).

Our pessimism about the usage of links under multipoint traffic does not increase the values of *maxusage* found. Whatever the value of *maxusage*(l) is, we can find a set of point-to-point requests which cause that much usage on link l.

It is not hard to see that the above defines a many to one correspondence from compatible request sets $\mathcal{R}$ to flows f in $\mathcal{L}_l$ (solutions x of LP_l). Furthermore, the value of f (objective value of x) is equal to the usage of l, $\sum_{q \in \mathcal{R}_l} rate(q)$.

6. Conclusion

We have presented a new way of specifying traffic in communication networks which we believe is easier for a network manager to specify and predict. It is useful for specifying traffic in a multirate multipoint connection-oriented network such as ATM networks.

Given that we want to configure networks which are nonblocking with respect to these traffic limits, we have designed an efficient algorithm to compute a lower bound on the cost of any such network. Experimental results on randomly generated instances show that hierarchical star networks are usually very close to optimal. We conjecture that they are indeed optimal for flat α, ω-bounded traffic, and have made progress in proving this conjecture by showing that star networks are optimal among all tree-shaped networks.

Our probabilistic result lends analytical support to the observation that star networks are likely to be as close as we want to optimal as the number of nodes grows.

Even if a network manager does not wish to build a tree-shaped network, the lower bound is still useful in assessing the cost of any configuration the manager does examine. We have developed algorithms for configuring nonblocking networks with arbitrary shape, given that fixed path routing is used to route connections.

There are several extensions to this work that we plan to examine. One reason that a manager would not want to install a tree-shaped network is that such a network is not tolerant to link or node failures. We will examine network configurations which are tolerant to single link or node failures, called two-connected networks. Grötschel et al have examined a similar problem [GMS92].

Another extension is to allow the manager to specify the traffic based on the locations of users and their terminals, give possible locations where switches may be installed, and then let the configuration algorithm select the cheapest configuration. The current work assumes that the switch locations are fixed.

Finally, the traffic limits of a network usually change over periods of time. We will examine the case where there is already an installed network, and the manager wants to modify the installation as cheaply as possible so as to satisfy the new traffic. For flat α, ω-bounded traffic, this problem is very similar to that of expanding the outside plant of a telephone central office. This problem has been treated extensively by Jack, Kai, Shulman, and others [JKS92a, JKS92b].

References

[AV79] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18:155–193, 1979.

[FF64] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton Univ. Press, Princeton, NJ, 1964.

[Fin91] J. Andrew Fingerhut. Designing communication networks with fixed or nonblocking traffic requirements. Technical Report WUCS-91-55, Washington University, St. Louis, Missouri, 1991.

[Fin92] J. Andrew Fingerhut. Algorithms for designing nonblocking communication networks with general topologies. Technical Memorandum WUCS-TM-92-05, Washington University, St. Louis, Missouri, 1992.

[GMS92] Martin Grötschel, Clyde L. Monma, and Mechthild Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, March–April 1992.

[GT88] Andrew V. Goldberg and Robert Endre Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, October 1988.

[GT90] Andrew V. Goldberg and Robert Endre Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, August 1990.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, March 1963.

[HR90] Torben Hagerup and Christine Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33(6):305–308, 1990.

[Hu69] Te Chiang Hu. *Integer Programming and Network Flows.* Addison-Wesley, 1969.

[JKS92a] Carolyn Jack, Sheng-Roan Kai, and Alexander Shulman. Design and implementation of an interactive optimization system for telephone network planning. *Operations Research*, 40(1):14–25, January-February 1992.

[JKS92b] Carolyn Jack, Sheng-Roan Kai, and Alexander Shulman. NETCAP - an interactive optimization system for GTE telephone network planning. *Interfaces*, 22(1):72–89, January-February 1992.

[KRT92] V. King, S. Rao, and Robert Endre Tarjan. A faster deterministic maximum flow algorithm. In *Proc. 3rd ACM-SIAM Symp. Discrete Algorithms*, pages 157–164, January 1992.

[Min89] Michel Minoux. Network synthesis and optimum network design problems: Models, solution methods and applications. *Networks*, 19:313–360, 1989.

[MT89] Riccardo Melen and Jonathan S. Turner. Nonblocking multirate networks. *SIAM J. Comput.*, 18(2):301–313, April 1989.

[MT90] Riccardo Melen and Jonathan S. Turner. Nonblocking multirate distribution networks. In *Proc. INFOCOM*, pages 1–8?, 1990.

[MW84] Thomas L. Magnanti and Richard T. Wong. Network design and transportation planning: models and algorithms. *Transportation Science*, 18(1):1–55, February 1984.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, 1982.

[Tar83] Robert Endre Tarjan. *Data Structures and Network Algorithms.* SIAM, 1983.